Universidad de Cuenca

Facultad de Ingeniería

Escuela de Electrónica y Telecomunicaciones & Escuela de Sistemas

Implementación de un Set-Top Box basado en plataformas de hardware de bajo costo

TESIS PREVIA A LA OBTENCIÓN DEL TÍTULO DE INGENIERO EN ELECTRÓNICA Y TELECOMUNICACIONES & INGENIERO DE SISTEMAS

Autores :

Yanathy Helena Inga Lojano

María del Cisne Romero Torres

Director :

Ing. Darwin Fabián Astudillo Salinas, PhD

Co-Director :

Ing. Kenneth Samuel Palacio Baus, MSc

Cuenca - Ecuador

2016

Resumen

En el año 2010 Ecuador adoptó el estándar de Televisión Digital *Integrated Services Digital Broadcasting, Terrestrial, Brazilian* (ISDB-Tb), sumándose a muchos otros países de Latinoamérica, generando oportunidades para el estudio y desarrollo de trabajos en este campo. En este contexto, surge la necesidad de promover el desarrollo de tecnologías que aprovechen las ventajas de transmisión de contenido de alta definición y aplicaciones interactivas con el fin de mejorar la experiencia del televidente.

La reciente introducción de esta tecnología abre oportunidades de investigación sobre el uso, transmisión y recepción de señales de *TV digital*(TDV), lo que implica analizar diferentes alternativas de hardware y software que posibiliten un alto índice de penetración y cobertura. Adicionalmente y con el objetivo de impulsar el cambio de la matriz productiva del país, este trabajo tiene como propósito implementar un dispositivo receptor de señales de TDV conocido como *Set Top Box* (STB), que integre una plataforma de hardware que soporte un componente de software, denominado *middleware*, orientado a facilitar la interactividad. Se abordan diferentes aspectos relacionados con el diseño e implementación de un STB sobre una plataforma popular y de bajo costo, entre ellos: arquitectura, configuraciones y pruebas.

Primero se presenta los fundamentos teóricos y la situación actual en Ecuador de la TDV, describiendo el estándar adoptado en el país (sistema de transmisión y recepción), arquitectura y funcionamiento de un STB, y su *middleware*. Posteriormente se propone la evaluación de diferentes plataformas de hardware: Arduino, Raspberry Pi y BeagleBone, sobre las cuales se describe sus características y uso; adicionalmente, se analizan distintos componentes de software compatibles con la plataforma de hardware seleccionada, mediante la comparación de viabilidad de uso de tres sistemas operativos de código abierto: Raspbmc, Raspbian y Ubuntu Mate. Además, se plantea un procedimiento para incorporar un dispositivo sintonizador externo que posibilite la recepción de señales ISDB-Tb. Una vez evaluada y seleccionada la plataforma de hardware, se implanta el *middleware* sobre esta plataforma, para finalmente evaluar el funcionamiento integral del sistema mediante la ejecución de aplicaciones de prueba. De esta manera se logra mostrar la factibilidad de implementar y evaluar un prototipo de STB de bajo costo.

Palabras clave : STB, ISDB-Tb, TDT, Middleware, Ginga .

Abstract

Ecuador adopted in 2010 the Digital Television standard known as ISDB-Tb, which has been chosen by many countries in Latin America generating research and development opportunities in a growing field. Therefore, it is important to promote and developt technologies which take advantage of high definition content transmition and also, the chance of improve viewer's experience through interactive applications.

This emerging field and its associated research opportunities require to analyze different hardware platforms and software alternatives which allow improving digital television (DTV) penetration indexes and coverage. Moreover, aiming to cooperate with the Ecuador's objective on propitiating a change in the production matrix by locally producing aggregated value products, our work is focused on implementing a DTV signals receiver device, also known as set-top-box STB. This device comprises hardware and software components, able to support what is named a *middleware* layer which enables interactivity. We focus con several topics related to the design and the implementation of a STB based on a low cost and popular platform, including architecture, configuration and testing.

First we present some theoretic principles and a summary of the current situation of DTV in Ecuador, by describing the technical aspects of the ISDB-Tb standard related to signal transmission and reception and the STB key components. Later, we propose an evaluation procedure aimed to test the following hardware components: Arduino board, Raspberry Pi and BeagleBone. These platforms are analyzed in detail in order to determine the one that best fits the STB minimum requirements, such that several software components can be tested for compatibility purposes, including Linux based operating systems such as Raspbmc, Raspbian and Ubuntu Mate. This work includes the development of a setup procedure that allows incorporating an ISDB-Tb DTV signal receiver to the selected hardware platform. Finally, a middleware component is incorporated to the system such that it can be thoroughly evaluated by making use of standard tests. This work shows that indeed a low cost platform based STB can be implemented and evaluated.

Keywords : STB, ISDB-Tb, TDT, Middleware, Ginga

Índice general

[1]https://www.arduino.cc/
[2]https://www.raspberrypi.org/
[3]http://beagleboard.org/bone

Índice de figuras

Índice de tablas

Yo, Yanathy Helena Inga Lojano, autor de la tesis "Implementación de un Set-Top Box basado en plataformas de hardware de bajo costo", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 05 de Abril de 2016

Yanathy Helena Inga Lojano
070587722-3

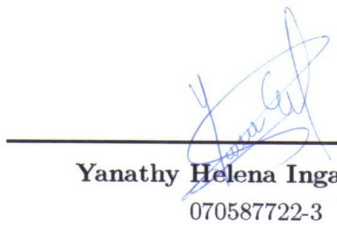
Yo, María del Cisne Romero Torres, autor de la tesis "Implementación de un Set-Top Box basado en plataformas de hardware de bajo costo", certifico que todas las ideas, opiniones y contenidos expuestos en la presente investigación son de exclusiva responsabilidad de su autor.

Cuenca, 05 de Abril de 2016

María del Cisne Romero Torres
070538151-5

Yo, Yanathy Helena Inga Lojano, autor de la tesis "Implementación de un Set-Top Box basado en plataformas de hardware de bajo costo", reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de ingeniero en Electrónica y Telecomunicaciones. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Cuenca, 05 de Abril de 2016



Yanathy Helena Inga Lojano
070587722-3

Yo, María del Cisne Romero Torres, autor de la tesis "Implementación de un Set-Top Box basado en plataformas de hardware de bajo costo", reconozco y acepto el derecho de la Universidad de Cuenca, en base al Art. 5 literal c) de su Reglamento de Propiedad Intelectual, de publicar este trabajo por cualquier medio conocido o por conocer, al ser este requisito para la obtención de mi título de ingeniero en Sistemas. El uso que la Universidad de Cuenca hiciere de este trabajo, no implicará afección alguna de mis derechos morales o patrimoniales como autor.

Cuenca, 05 de Abril de 2016

María del Cisne Romero Torres
070538151-5

Dedicatoria

Con todo mi amor y cariño a las personas que dedicaron su vida a la mía, a ustedes padres, por todo el esfuerzo que han realizado por mí, este logro es suyo. A mis mejores amigos, mis hermanos, por ser mi ejemplo, estar siempre a mi lado, y por regalarme la dicha de ser tía, para mis sobrinas también va dedicado este trabajo. A mis amigos, por esos abrazos sinceros, por las palabras, y silencios en los momentos adecuados. Al futuro, ya que siempre se llega a alguna parte si se camina lo suficiente.

Yanathy Helena

A las personas importantes en mi vida, con todo mi cariño dedico esta tesis a mis padres por ser el pilar fundamental en todo lo que soy, por sus consejos, sus valores; a ustedes y mis hermanos por haberme apoyado en todo momento y alentarme siempre a continuar. A Dios, por haberme permitido llegar hasta este punto y haberme dado salud para lograr mis objetivos. A mis compañeros, a mis maestros y amigos, quienes sin su ayuda nunca hubiera podido hacer esta tesis.

María del Cisne

Agradecimientos

Agradezco a Dios por guiarme y permitirme culminar esta etapa tan importante en mi vida. A mis padres, hermanos, y familiares por su apoyo incondicional en el transcurso de mi formación profesional. Al Ing. Fabián Astudillo, e Ing. Kenneth Palacio por su tiempo invertido, y motivación durante el desarrollo de este proyecto. A todos los profesores que compartieron sus conocimientos y nos brindaron su amistad, lo que fue indispensable para culminar con éxito esta carrera universitaria. A mi compañera de tesis, hermana, gracias por todo el apoyo incondicional y amistad sincera. Y a mis compañeros de carrera con los que hemos compartido buenos momentos e inolvidables.

Yanathy Helena

Quiero agradecer a mis padres, hermanos y a todos aquellos que participaron directa o indirectamente en la elaboración de este proyecto. A mi compañera Yana, gracias por tu amistad y porque juntas lo logramos. A mis directores de tesis por su paciencia y su motivación para que pueda terminar esta etapa. Son muchas las personas que han formado parte de mi vida profesional a las que me encantaría agradecerles su amistad, consejos, apoyo, ánimo y compañía en los momentos más difíciles. Gracias a ustedes, se los agradezco desde el fondo de mi alma.

María del Cisne

Abreviaciones y Acrónimos

AAC *Advanced Audio Coding*. 46
ADC *Analog Digital Converter*. 21
AMD *Advanced Micro Devices*. 21
APL *Arduino Programming Language*. 35
ARM *Acorn RISC Machine*. 21, 36, 46
ATSC *Advanced Television Systems Committee*. 8
AVC *Advanced Video Coding*. 10

BTS *Broadcast Transport Stream*. 11

CF *Canal Físico*. 13
CITDT *Comité Interinstitucional Técnico de Implementación de la Televisión Digital Terrestre*. 2
CMOS *Complementary metal–oxide–semiconductor*. 22
COFDM *Coded Orthogonal Frequency Division Multiplexing*. 9, 20
CPU *Central Proccessing Unit*. 20, 22, 36, 46
CV *Canal Virtual*. 13

DAC *Digital-to-analog converte*. 21
DIUC *Dirección de Investigación de la Universidad de Cuenca*. 2
DQPSK *Differential Quadrature Phase Shift Keying*. 12
DSM-CC *Medios de Almacenamiento Digital – Comandos y Control*. 13
DSP *Digital Signal Processor*. 21
DTMB *Digital Terrestrial Multimedia Broadcast*. 8
DVB-C *Digital Video Broadcasting - Cable*. 8
DVB-S *Digital Video Broadcasting - Satellite*. 8, 29
DVB-SI *Digital Video Broadcasting-Service Information*. 20
DVB-T *Digital Video Broadcasting - Terrestrial*. 8, 26–28, 31, 49

EDTV *Enhanced Definition Television*. 8

EEPROM *Electrically Erasable Programmable Read-Only*. 21
EIT *Tabla de Información Eventual*. 13
EPG *Guía electrónica de programación*. 3, 13, 18
ES *Elementary Stream*. 11, 13

FEC *Forward Error Correction*. 52
FI *Frecuencia Intermedia*. 12
FM *Frecuencia modulada*. 49
FPS *frames per second*. 10, 26

GPU *Graphics Proccessing Unit*. 18, 21, 22

HDMI *High-Definition Multimedia Interface*. 36, 41
HDTV *High Definition Television*. 8–11

IPTV *Internet Protocol Television*. 4, 16
ISDB-T *Integrated Services Digital Broadcasting, Terrestrial*. 8, 10
ISDB-Tb *Integrated Services Digital Broadcasting, Terrestrial, Brazilian*. 2–5, 7, 10, 13, 14, 16, 20, 31, 49

LCD *Liquid Crystal Display*. 3

MINTEL *Ministerio de Telecomunicaciones*. 14
MIPS *Microprocessor without Interlocked Pipeline Stages*. 21
MMU *Memory Manager Unit*. 21
MPEG *Moving Pictures Experts Group*. 20, 46

NCL *Nested Context Language*. 4, 30, 61–63
NCM *Nested Context Model*. 61
NIT *Tabla de información de Red*. 12

PAT *Tabla de Asociación de Programa*. 12
PCI *Peripheral Component Interconnect*. 17, 29
PES *Packetized Elementary Stream*. 11, 13
PID *Packet Id*. 17
PLL *Phase-Locked Loop*. 21
PMT *Tabla de Mapa de Programa*. 12
PoR *Power-on Reset*. 21
PPV *Pay Per View*. 16, 18

QAM *Quadrature Amplitude Modulation*. 9, 12
QPSK *Quadrature Phase Shift Keying*. 9, 12

RAM *Random Access Memory*. 16, 21, 36, 41, 46
RF *Radiofrecuencia*. 17, 22, 50
ROM *Read Only Memory*. 19, 21
RTE *Reglamentos Técnicos Ecuatorianos*. 4, 13

SBTVD-T *Sistema Brasileiro de Televisión Digital*. 8
SDTV *Standard Digital Television*. 8–11
SFN *Single Frequency Networks*. 9
SIP *System in Package*. 7, 22
SoC *System-on-a-chip*. 7, 21, 22, 30, 31, 36, 38, 46
SOP *System on Package*. 22
SPI *Serial Peripheral Interface*. 21
STB *Set Top Box*. 3–8, 13, 14, 16, 18–22, 25–27, 29–31, 45–49, 51, 59, 60, 62, 67, 70

TDT *Televisión Digital Terrestre*. 3–6, 8–11, 13–15, 18, 27, 30
TS *Transport Stream*. 11–13, 17, 20
TSP *Transport Stream Packet*. 11

UHF *Ultra High Frecuency*. 9, 15, 17, 50
USART *Universal Synchronous/Asynchronous Receiver/Transmitter*. 21
USB *Universal Serial Bus*. 21, 34, 49–51, 54, 57, 71

VHF *Very High Frequency*. 9, 17, 50
VoD *Video on demand*. 16

XML *eXtensible Markup Language*. 61, 63

Capítulo 1

Introducción

En este capítulo se aborda los temas relacionados con la televisión digital en el Ecuador y como ha venido trabajando en este contexto la Universidad de Cuenca. Se presenta además la identificación del problema, justificación, alcance y objetivos de este proyecto de tesis. En la problemática se expone la escasez de receptores que cumplan los requerimientos establecidos en el país, el escaso desarrollo e investigación en el ámbito tecnológico, como equipos que permitan la interactividad con el usuario y funciones extras con el menor costo posible; dado este enfoque se justifica la adecuación de un equipo receptor. Finalmente, se muestra el alcance del proyecto en donde se especifica las etapas a estudiar para cumplir con los objetivos expuestos.

1.1. Presentación

El 26 de marzo de 2010, Ecuador adoptó oficialmente el estándar japonés-brasileño ISDB-Tb para la Televisión Digital Terrestre [20]. Desde la fecha indicada hasta la actualidad se ha probado la transmisión de señales de televisión digital en algunas provincias del Ecuador.

Según el *Comité Interinstitucional Técnico de Implementación de la Televisión Digital Terrestre* (CITDT), 23 operadores de televisión abierta de los 89 autorizados a nivel nacional transmiten señales de televisión en formato digital en las ciudades de Quito, Guayaquil, Cuenca, Santo Domingo, Manta, Latacunga y Ambato.

Se ha planificado que en los próximos tres años, todos los operadores puedan transmitir permanentemente la señal de TV digital, lo que se conoce como el apagón analógico que está previsto iniciar en el 2016 y terminar en el 2018. Hasta la fecha, se pretende que exista suficiente cobertura en la mayor parte del territorio ecuatoriano y de ésta forma lograr que la mayor parte de la población tenga acceso a este servicio y disponga de las ventajas de la televisión digital.

En resumen, dadas la señal de televisión analógica y digital, existe un cambio sustancial en la manera de representación de los datos, gracias a lo cual la calidad de servicio en relación a la televisión analógica ha mejorado significativamente; además tiene como complemento la interacción entre el usuario y los proveedores de difusión, esto quiere decir que el usuario estará en la capacidad de comunicarse mediante aplicaciones en varios ámbitos vía canal de retorno, por ejemplo puede revisar propuestas para hacer compras en línea, revisar la meteorología, votar y/o apostar en concursos, etc.

Dentro de este escenario, la Universidad de Cuenca ha venido desarrollando proyectos de investigación sobre este tema, se inició con el proyecto aprobado y financiado por la *Dirección de Investigación de la Universidad de Cuenca* (DIUC) denominado: “Aplicación de Tecnologías Semánticas para Disminuir la Sobrecarga de Información en Usuarios de TV digital”, que pretende diseñar un sistema de recomendación para la programación televisiva de acuerdo a las preferencias del usuario [9]. Este proyecto se ha dividido en dos grandes etapas como se observa en la Figura 1.1.

La primera fase del proyecto, y en la cual se enfocará esta investigación, consiste en el “Laboratorio de TV digital” la cual, en la primera etapa se encarga de buscar las alternativas óptimas para simular un escenario real de transmisión y recepción de señal televisiva, aspectos que se abordaron en el trabajo de Arturo Gonzalo Gutiérrez Tapia y Miguel Ángel Cochancela Alvear “Diseño de un laboratorio de televisión digital para la transmisión de señales con multiprogramación, contenidos interactivos y EPG” [10]. Como resultado, los autores obtuvieron varias alternativas tanto en hardware como en software para la implementación y adecuación

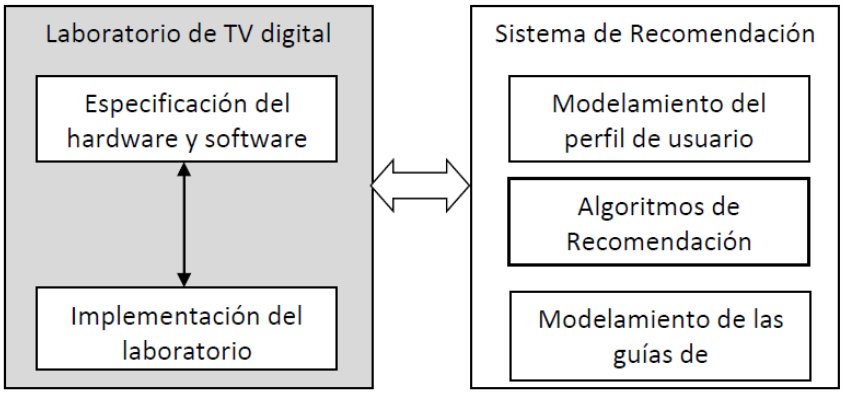


Figura 1.1: Esquema general del proyecto "Aplicación de Tecnologías Semánticas para Disminuir la Sobrecarga de Información en Usuarios de TV digital", Fuente: [9, Fig. 1]

del laboratorio de televisión digital.

La segunda etapa consistió en la selección de una de dichas alternativas y su respectiva implementación. Este trabajo fue elaborado por José Luis Medina Cartuche y Cristian Ramiro Villa Arias "Implementación de un laboratorio de televisión digital que cumpla el estándar ISDB-Tb" [9], como resultado, fue equipado un laboratorio con elementos de hardware y software, los cuales posibilitan transmitir y recibir señales en formato digital [9, Cap. 2]. La arquitectura del laboratorio se representa en la Figura 1.2, en la que se distingue 4 módulos destinados a una función específica del laboratorio.

- Estación de generación de señales (*Broadcaster*).
- Ambiente de desarrollo de aplicaciones.
- Proveedor de servicios interactivos (*Return Channel Server*).
- Entorno del usuario o receptor.

Al final de la investigación se encontraron temas adicionales que no se profundizaron, por ejemplo, una de las cosas que resta investigar es diseñar un STB propio que permita incorporar aplicaciones interactivas empotradas que faciliten la interacción del usuario con el *broadcaster*, esta tarea es parte del proyecto de investigación "Empleo de Tecnologías Semánticas para el Análisis de Contenido Multimedia transmitido para Televisión Digital Terrestre".

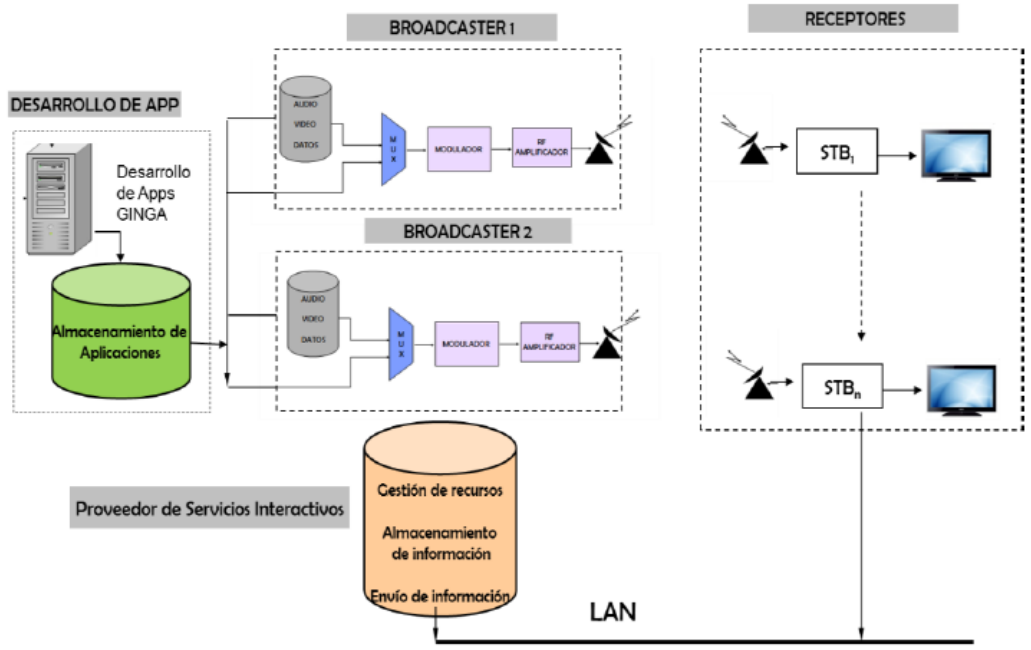


Figura 1.2: Diseño de arquitectura del laboratorio, Fuente: [9, Fig. 6]

1.2. Identificación del Problema

Mediante la resolución No. 084-05-CONATEL-2010 del 25 de Marzo de 2010 [20], Ecuador se acoge a la tendencia mundial de cambio hacia la *Televisión Digital Terrestre* (TDT), con la adopción del estándar japonés-brasileño ISDB-Tb. Existen dos componentes principales dentro de la arquitectura, el *broadcaster* (transmisor) y los receptores, este trabajo se enfoca en el segundo componente, donde se tendrá dos escenarios en el período de transición hacia la TDT. Como primer escenario, los usuarios adquieren televisores nuevos, en los cuales el decodificador ISDB-Tb está integrado; y, el segundo escenario, en donde los usuarios deciden continuar usando los televisores con recepción analógica o LCD-Plasmas importados en el pasado, los cuales no son compatibles con ISDB-Tb, y tendrían que conectar un receptor digital externo (STB).

El 23 de diciembre de 2013, se emitió el Reglamento Técnico RTE 83 para la importación, fabricación, ensamblaje y comercialización de televisores, en el que se establece que deben ser aptos para la recepción de televisión digital usando el estándar ISDB-Tb [21]. De esta manera en unos años habrá un punto de inflexión en donde se supere la escasez actual de televisores con este receptor en el mercado nacional. Aunque el mencionado reglamento establece el uso de ISDB-Tb, no obliga a que los televisores tengan el *middleware* Ginga instalado.

Por otro lado, el enfoque del trabajo a realizar va orientado a resolver el inconveniente de los

televisores con recepción analógica. El problema radica en el escaso desarrollo e investigación del país en el ámbito tecnológico, ya que no hay equipos STB de fabricación nacional que soporten el estándar ISDB-Tb y el *middleware* Ginga. Como una primera aproximación para resolver el problema antes mencionado, este trabajo busca implementar un STB que soporte Ginga, integrando componentes de bajo costo que existen en el mercado nacional.

1.3. Justificación

Actualmente el laboratorio de televisión digital de la Universidad de Cuenca, cuenta con equipamiento necesario para la generación, transmisión, y recepción de señal digital televisiva. Para cumplir con el objetivo, se adquirió un decodificador ISDB-Tb modelo: *EiTV Developer ISDB-T e IPTV* con una herramienta de virtualización basada en un servidor Linux. Este decodificador está destinado para desarrolladores, y entre sus funciones permite recibir y decodificar *Internet Protocol Television* (IPTV) y señales de TDT con el estándar ISDB-Tb. También permite ejecutar aplicaciones Ginga con librerías Lua por medio de una máquina virtual instalada en el equipo.

Sin embargo, existieron inconvenientes en las pruebas realizadas por los autores de [9], donde tuvieron que adecuar la aplicación desarrollada en NCL, que se basó en el registro de usuarios y un sistema recomendador, dicha aplicación presentó incompatibilidad del equipo con algunas librerías de Lua, que es un lenguaje universal adaptado en Ginga, que en este caso sirve para la captura de datos: como número y nombre del canal, y nombre del programa. A pesar de que el dispositivo *EiTV Developer ISDB-T e IPTV* presenta características de un equipo de desarrollo, cuenta con capacidades muy limitadas en cuanto al procesamiento y ejecución de las aplicaciones.

En la actualidad, en el país se cuenta con escasos equipos receptores que tengan el *middleware* Ginga incluido. Por tal razón este proyecto está dedicado a la implementación de un STB con la integración de plataformas de hardware de bajo costo, donde se obtenga compatibilidad con el *middleware*. Este trabajo se realizará de manera multidisciplinaria, integrando las carreras de: Electrónica y Telecomunicaciones, y Sistemas.

Electrónica y Telecomunicaciones aborda aspectos como: análisis de hardware necesario para la implementación de un STB. La implementación se divide en dos etapas: sintonización de la señal y procesamiento. Para la etapa de sintonización se requiere el análisis de varios sintonizadores (*dongles*) existentes en el mercado mediante el estudio de la arquitectura, características de recepción (*one-seg, full-seg*), soporte del estándar ISDB-Tb y soporte de las bandas de 174 a 698 MHz. Luego del análisis se procede a realizar las pruebas de recepción mediante consola

y analizar los datos obtenidos. El aporte de Sistemas está enfocado a la evaluación de sistemas operativos, compilación del kernel para el correcto acoplamiento de los dispositivos externos, implantación del *middleware* Ginga desarrollado en Brasil y desarrollado en Argentina, mediante la compilación del código, tomando en cuenta las librerías y dependencias necesarias y modificación de archivos. Así como la evaluación para determinar cuál se adapta mejor al sistema, tomando en cuenta sus limitaciones y ventajas. Obteniendo como resultado un conjunto de procedimientos que permitan la evaluación completa del STB, garantizando un entorno que ofrezca aceptación por parte del usuario.

1.4. Alcance

Este proyecto de tesis tiene como finalidad implementar un STB que tome en cuenta los requerimientos en Ecuador. Se realizará un estudio de trabajos realizados que servirán como apoyo para la investigación. En una primera etapa se identificarán los requerimientos mínimos de un STB (velocidad de procesamiento, memoria, compatibilidad con SO, interfaces, sintonización, decodificación), por lo que se plantean varias alternativas de plataformas de hardware y software; y, de acuerdo a los requerimientos anteriores se determinará cual tiene mejores características y cual se acoplaría al *middleware* Ginga, para cumplir con el objetivo propuesto.

En el transcurso de la implementación de dicha solución se hará el análisis de elementos adicionales de hardware necesario para tener como resultado un STB dedicado. De esta manera, se plantea procedimientos para el acople entre dispositivos, tanto para la sintonización como visualización de la TDT. Además, de aprovechar los recursos de hardware de la plataforma para mejorar el procesamiento de las señales sintonizadas y obtener un sistema eficaz para el usuario.

Finalmente, se agregará el *middleware* a la plataforma propuesta. De acuerdo al estándar adoptado en Ecuador, el *middleware* a ser usado es Ginga [1]. Se tiene varias versiones y desarrolladores, como: Ginga desarrollado en Brasil y Ginga desarrollado en Argentina; para determinar la mejor opción se estudiará la compatibilidad con el hardware y el sistema operativo. Luego, se instalará mediante compilación de código, y se comprobará su funcionamiento mediante la ejecución de una aplicación interactiva.

[1] http://www.ginga.org.br y http://tvd.lifia.info.unlp.edu.ar

1.5. Objetivos

1.5.1. Objetivo General

Implementar un STB basado en plataformas de hardware de bajo costo, que cumpla los requerimientos de recepción del formato de televisión digital adoptado en el Ecuador, y que posibilite la ejecución de aplicaciones interactivas basadas en Ginga.

1.5.2. Objetivos específicos

- Recopilar trabajos relacionados con el diseño de STBs.
- Proponer un procedimiento que permita evaluar diferentes dispositivos para la recepción de la TDT y para el procesamiento de la señal utilizando ordenadores de bajo costo.
- Implementar un STB mediante la integración de las plataformas de hardware seleccionadas que permita la recepción de una señal de TV digital.
- Proponer un procedimiento que permita evaluar la compatibilidad de la implantación del *middleware* Ginga de desarrollo brasileño y argentino sobre la plataforma seleccionada.

Capítulo 2

Marco Teórico

En el presente capítulo se definen los conceptos necesarios para este proyecto de tesis. Se inicia con una presentación sobre la televisión digital, su descripción, las ventajas que brinda, y su estado actual en el Ecuador. Luego se describe el estándar ISDB-Tb adoptado en el país: las características y arquitectura del sistema de transmisión y recepción. Y finalmente, se detalla las características, arquitectura y funcionamiento de un STB.

2.1. Televisión Digital Terrestre

La TDT es el resultado de la aplicación de la tecnología digital a la señal de televisión, ofrece video en alta definición, división de múltiples sub-canales y contenidos digitales adicionales, se puede sintonizar en un televisor que tenga una antena y un STB incorporado o un STB externo.

Presenta diferentes niveles de calidad y formato de señal: *Standard Digital Television* (SDTV), *Enhanced Definition Television* (EDTV), *High Definition Television* (HDTV). Existen tres formas de transmisión de TDT: DVB-S, *Digital Video Broadcasting - Terrestrial* (DVB-T), DVB-C. Además cinco conjuntos de tecnologías o estándares:

- *Advanced Television Systems Committee* (ATSC), sistema Americano.
- DVB-T, sistema europeo.
- *Integrated Services Digital Broadcasting, Terrestrial* (ISDB-T), sistema japonés.
- *Digital Terrestrial Multimedia Broadcast* (DTMB), sistema chino.
- *Sistema Brasileiro de Televisión Digital* (SBTVD-T), sistema brasileño basado en el japonés.

Estos estándares usan diferentes métodos de modulación, codificación, transmisión y recepción, con el fin de tener mayores ventajas con respecto a la televisión analógica, como: uso más eficiente del espectro, mejor calidad de servicio, servicios adicionales como la interactividad. Las estaciones transmisoras tendrán que modernizar los equipos para la generación de contenidos en formato digital lo cual tendrá un alto costo, sin embargo, este costo será amortizado gracias a las ventajas antes mencionadas.

2.1.1. Ventajas

A continuación se describen algunas ventajas de la TDT.

a. Mejor calidad de imagen y sonido

La señal TDT no presenta los problemas de recepción que tiene la señal analógica, como son: ruido, colores deficientes, doble imagen y sonido de baja calidad. Debido a la presencia de ruido, e interferencias la señal analógica reduce notablemente su calidad. En la TV digital al tratarse la señal mediante técnicas de modulación y codificación, el receptor puede corregir hasta cierto punto las distorsiones provocadas por interferencias. Sin embargo, cuando el receptor no es capaz de reparar ciertos errores puede producirse la congelación de partes de la imagen o la interrupción del sonido; y cuando el nivel de

error supera cierto umbral, se muestra una imagen negra y sin sonido.

En la TDT la resolución de imagen aumenta con respecto a la resolución de la televisión analógica, ofrece varios formatos de calidad de imagen: Full HDTV con resolución de 1920x1080 pixeles; HDTV con 1280x720 pixeles y SDTV con una resolución de 720x480 pixeles; mientras que la televisión analógica solo ofrece una resolución de 640x480 píxeles. En relación a la calidad de audio, en la televisión analógica la señal se transmite por medio de canales estéreo (izquierdo y derecho), mientras que en la TDT permite enviar hasta seis canales de audio (izquierdo, derecho, central, *surround* izquierdo, *surround* derecho y la sexta señal que es una extensión de baja frecuencia llamada *subwoofer*), lo cual mejora la calidad del audio.

b. Optimización del espectro radio eléctrico

Permite usar de forma más eficiente el espectro radioeléctrico y transmitir más de una señal en un mismo ancho de banda, a diferencia de la señal analógica que puede transmitir una; ésto se da gracias al uso de técnicas de modulación en la señal digital como son: *Quadrature Phase Shift Keying* (QPSK), 16*Quadrature Amplitude Modulation* (QAM) y 64QAM; ya sea en la banda de *Very High Frequency* (VHF) o *Ultra High Frecuency* (UHF), y a la utilización de *Coded Orthogonal Frequency Division Multiplexing* (COFDM), ésta ultima es una técnica compleja de modulación para transmitir información digital a través de un canal de comunicaciones, acopla varios métodos de codificación para la corrección de errores en el receptor. COFDM modula la información en múltiples frecuencias portadoras ortogonales donde cada una está modulada en amplitud y fase, y lleva una tasa de símbolos muy baja, además de tener una alta eficiencia espectral. Adicionalmente, se podría transmitir la señal a través de *Single Frequency Networks* (SFN), lo que permite usar la misma frecuencia para estaciones transmisoras y repetidoras.

c. Servicios adicionales

La tecnología TDT permite enviar mayor información por el mismo canal. Se puede transmitir varios programas por cada canal UHF usando 6 MHz, 7 MHz u 8 MHz de ancho de banda. Además, la capacidad de transmisión se puede usar de manera flexible, se puede enviar una señal Full HDTV; o una señal HDTV y un SDTV, o 3 canales SDTV. Los servicios que ofrece la TDT son: canales HD, mayor número de canales, guía de programación, video vigilancia, canales de radio, teletexto digital. Asimismo, se ofrece servicios interactivos tales como: votaciones y encuestas, información de servicios públicos (tráfico, aeropuertos, meteorología), entre otros.

d. Movilidad y portabilidad

La TDT tiene una señal para dispositivos móviles llamada transmisión *one-seg*. La porta-

bilidad es la posibilidad de ver contenidos en dispositivos que se encuentren en movimiento, como equipos incorporados en buses, autos y trenes. La señal de recepción móvil es de menor calidad que la de recepción fija debido a que tiene menor velocidad de transmisión, sin embargo, ésto no es necesariamente una desventaja, ya que para las características de los dispositivos mencionados anteriormente es suficiente.

2.2. Estándar ISDB-Tb

ISDB-T Internacional, ISDB-Tb o SBTVD-T es un estándar para transmisión de TDT desarrollado en Brasil. Este estándar está basado en la norma japonesa ISDB-T [22] con algunas diferencias. La diferencia principal entre ISDB-T y el ISDB-Tb es la utilización de compresión de video, ISDB-T emplea H.262/MPEG-2 e ISDB-Tb H.264/MPEG-4 y *Advanced Video Coding* (AVC). Asimismo, permite una velocidad de 30 FPS para las transmisiones a dispositivos móviles a diferencia del ISDB-T que está limitado a 15 FPS.

2.2.1. Estructura del Sistema TDT

La estructura TDT se compone del sistema de transmisión y recepción, se expone el tema de banda segmentada, un diagrama de bloques del sistema en general, el ancho de banda, y tipos de canales.

2.2.1.1. Transmisión en banda segmentada

El sistema ISDB-Tb permite organizar la información a transmitir en tres capas jerárquicas diferentes: A, B y C, para servicios *one-seg*, SDTV, y HDTV respectivamente. El canal se ha dividido en 13 segmentos, convirtiéndolo en un sistema de banda segmentada, reservando un segmento para la banda de guarda, por lo tanto, el ancho de banda de cada segmento se rige bajo la siguiente ecuación: [10]

$$Bws = \frac{Bwc}{14} = \frac{600KHz}{14} = 428,5714KHz \tag{2.1}$$

Donde *Bwc* representa el Ancho de Banda de canal, y *Bws* el Ancho de Banda de un segmento.

Figura 2.1: Segmentación del canal - Estándar ISDB-Tb, Fuente: [10, Fig. 3.1]

La Figura 2.1 muestra la segmentación del canal. El segmento central siempre se lo reserva para transmitir *one-seg*, y los segmentos restantes son utilizados para transmitir un Full HDTV, o un HDTV con un SDTV o bien tres programas SDTV.

2.2.1.2. Diagrama de bloques del sistema TDT [1]

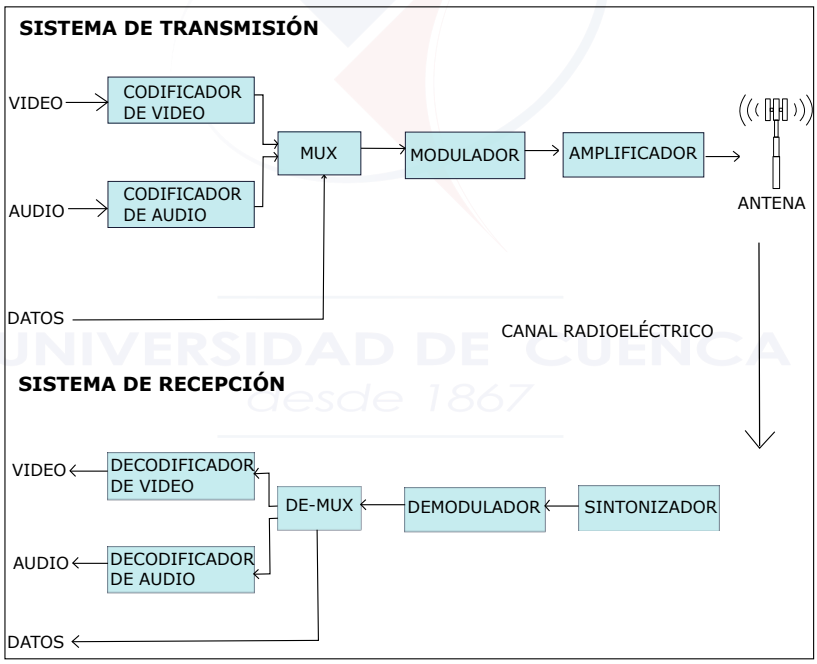


Figura 2.2: Sistema TDT estándar ISDB-Tb

El sistema TDT está formado por el sistema de transmisión y sistema de recepción (Figura 2.2). La información de video, audio y datos son comprimidos y multiplexados para formar un *Elementary Stream* (ES). Los ES son empaquetados para obtener a la salida un *Packetized Elementary Stream* (PES). Los cuales son multiplexados de nuevo con los datos de origen de otros programas para formar un *Transport Stream* (TS). El TS consiste en paquetes de

transporte de longitud fija de 188 bytes. Opcionalmente en un segundo nivel se multiplexa los TS para obtener un *Broadcast Transport Stream* (BTS) con las siguientes características:

- Se forma nuevos paquetes TSP, los cuales serán la información binaria total a transportar, cuya longitud es de 204 bytes (188+16), los 16 bytes son nulos, para poder mantener la velocidad binaria constante e independiente de los parámetros de transmisión seleccionados para cada capa jerárquica.
- El flujo es sincrónico y tiene una tasa constante de 32,5079 Mbps.
- Posibilita la transmisión jerárquica y la recepción parcial (*one-seg*).
- La cantidad de paquetes en cada capa varía, dependiendo de la configuración de transmisión.

Durante la transmisión se generan varios errores debido a la interferencia del medio, el ruido, desvanecimiento, etc. por lo que en la etapa de codificación se introducen algunos algoritmos a la señal para ayudar al receptor a la detección y corrección de errores (Algoritmo Reed Solomon). Luego se realiza un proceso de modulación. Donde la señal es convertida para la frecuencia del canal de transmisión y sometida al amplificador de potencia. Las frecuencias centrales de los canales digitales deben ser movidas 1/7MHz o 142,857kHz en relación al centro del canal, para evitar interferencias, este proceso se lo denomina decalaje de frecuencia, como se muestra en la Figura 2.3.

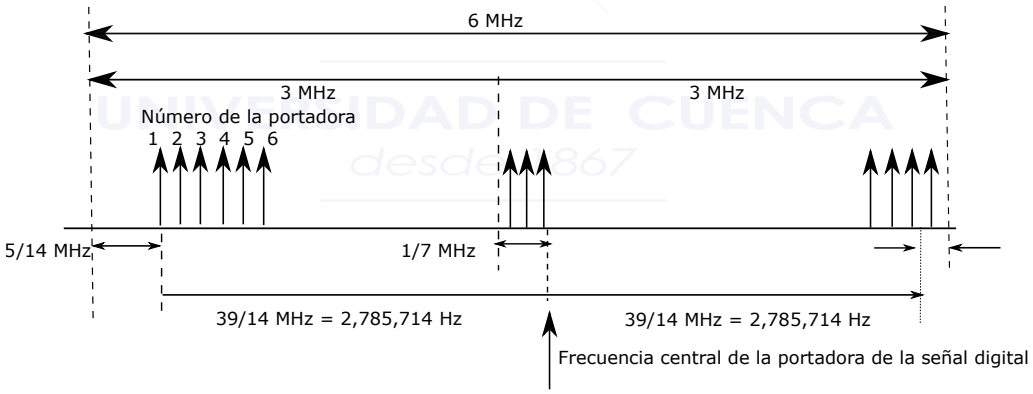


Figura 2.3: Decalaje de frecuencia de canal, Fuente: [11, Fig. 8.7]

El sistema de recepción realiza la secuencia de operaciones en orden inverso a lo que se realizó en el lado de transmisión, como se muestra en la Figura 2.2, la señal recibida por la antena pasa a etapa de amplificación para ser convertida desde la frecuencia del canal sintonizado a un valor de *Frecuencia Intermedia* (FI) de 44 MHz. Luego, se procede a la conversión de la señal analógica a digital, de esta manera se recuperan los símbolos, y se extrae la información para

la organización y configuración de las capas jerárquicas de acuerdo al esquema de modulación correspondiente (QPSK, *Differential Quadrature Phase Shift Keying* (DQPSK), 16-QAM, 64-QAM), recuperando las secuencias de bits.

A continuación los errores de la secuencia de bits son corregidos y se recupera la organización por bytes, finalmente en la etapa de decodificación se recupera el TS estructurado en paquetes de 188 bytes, que contiene los distintos programas o servicios y datos de las tablas.

Luego de la demultiplexación, se recupera la tabla NIT del servicio, las tablas PAT y PMT que permiten obtener las direcciones de los paquetes de audio y video que corresponden al programa elegido. Esta información se entrega al decodificador, que convierte las secuencias en paquetes PES y luego en flujos ES, aplicando los procesos que permiten obtener el audio y el video original.

El demultiplexor entrega los datos de las tablas EIT que transporta la EPG y la información *Medios de Almacenamiento Digital – Comandos y Control* (DSM-CC); que constituye el canal de datos y se utiliza para transmitir las aplicaciones interactivas que serán almacenadas en el STB. Las aplicaciones se cargan sobre el *middleware* Ginga, y de esa forma se consigue que la electrónica del receptor pueda interpretar los códigos de las aplicaciones.

Por otra parte es necesario conocer las características de ancho de banda y sensibilidad de recepción. El ancho de banda se especifica según el tipo de dispositivos empleados. Para dispositivos fijos o móviles de recepción *full-seg* se cuenta con 5.7 MHz; y, para dispositivos portátiles de recepción *one-seg* se cuenta con 0.43 MHz. En cuanto a la sensibilidad de recepción las unidades de sintonía para receptores de 13 y 1 segmentos, deben cumplir con las siguientes características: nivel de la señal de -20 dBm o superior, nivel mínimo de entrada de señal de la antena de -77 dBm o inferior, y nivel de entrada para receptor *one-seg* de -11 dB [23].

Existen dos tipos de canales: *Canal Físico* (CF) que se denomina a la frecuencia real de la portadora con todos los servicios complementarios dentro de la banda de frecuencia de 6 MHz. Y *Canal Virtual* (CV) que se refiere al número del canal en el cual el receptor muestra la programación de una estación TDT, independientemente del canal físico en el cual se transmite. La configuración de CV se encuentra embebida en el TS que se transmite a todas las estaciones repetidoras del sistema [23].

2.3. Situación Actual de la TDT en Ecuador

El 26 de marzo de 2010 se adoptó de manera oficial en el Ecuador el estándar japonés-brasileño ISDB-Tb, para la TDT. De esta manera el país entero contará con el estándar adop-

tado por la mayor parte de los países de Latinoamérica. La TDT fue adoptada por las funcionalidades y características que brinda, como son: mayor calidad en la imagen, movilidad, portabilidad, multiservicios, optimización del espectro e interactividad. Además, el gobierno promueve el acceso al servicio de televisión abierta de manera libre y gratuita; con la generación de contenidos de educación, salud y cultura que mejoren la calidad de programación.

El 23 de diciembre de 2013 se emitió el Reglamento Técnico RTE 83 para Televisores [21] para evitar que se comercialicen equipos que no cumplan con el estándar adoptado en el país. Este reglamento establece que todos los televisores, que se importen, fabriquen, ensamblen o comercialicen en el Ecuador a partir de la fecha antes mencionada, deben ser aptos para el estándar ISDB-Tb. Pero en caso de contar con un televisor convencional (no cuenta con ISDB-Tb) se podrá observar los canales de TDT incorporando un decodificador STB.

Según estadísticas del *Ministerio de Telecomunicaciones* (MINTEL) realizadas en el 2013, se requiere digitalizar 519 estaciones de TV abierta, de las cuales 16 % son matrices y el 84 % repetidoras [19]. Actualmente se emiten señales de prueba de TDT en varias ciudades del país; en la Tabla 2.1 se detallan las operadoras que se encuentran emitiendo TDT.

N°	Nombre de la Estación	M (Matriz)/ R (Repetidor)	Canal Virtual	Área Servida
1	ECUADOR TV	M	7	QUITO
2	TELEVISIÓN DEL PACIFICO	M	2	QUITO
3	TELEAMAZONAS	M	4	QUITO
4	TELESISTEMA	M	5	QUITO
5	TELEVISORA NACIONAL	M	8	QUITO
6	TELEVISIÓN SATELITAL	M	25	QUITO
7	TELESUCESOS	M	29	QUITO
8	46 UHF ABC	M	46	QUITO
9	CANAL UNO	M	12	QUITO
10	ECUADOR TV	R	7	GUAYAQUIL
11	CORPORACIÓN ECUATORIANA DE TELEVISIÓN	M	2	GUAYAQUIL
12	RED TELESISTEMA (R.T.S)	M	4	GUAYAQUIL
13	TELEAMAZONAS GUAYAQUIL	M	5	GUAYAQUIL
14	CADENA ECUATORIANA DE TELEVISIÓN	M	10	GUAYAQUIL

15	CANAL UNO	M	12	GUAYAQUIL
16	TV+ (TEVEMAS)	M	26	GUAYAQUIL
17	TELEVISIÓN SATELITAL	M	36	GUAYAQUIL
18	COSTANERA (RTU)	M	30	GUAYAQUIL
19	ECUADOR TV	R	7	CUENCA
20	UNIMAX	M	34	AMBATO-LATACUNGA
21	COLOR TV	M	36	AMBATO-LATACUNGA
22	OROMAR	M	41	MANTA-PORTOVIEJO
23	TELEATAHUALPA	M	25	SANTO DOMINGO

Tabla 2.1: Canales digitales de prueba en Ecuador, Fuente: [19]

2.3.1. Bandas de frecuencias

Según el informe No. CITDT-GATR-2012-005 [24], del 11 de septiembre del 2012, elaborado por el Grupo de Aspectos Técnicos y Regulatorios, expone la metodología de asignación de frecuencias temporales para la operación de estaciones de TDT que define: La banda de frecuencia que se usará para la transmisión de TDT es la banda UHF del espectro radioeléctrico, atribuida para el servicio de radiodifusión con emisiones de televisión.

Las bandas y canales para la implementación de la TDT en el Ecuador, se describen en la Tabla 2.2.

BANDA (MHz)	CANALES
174 - 216	7 - 13
470 - 482	14 - 15
512 - 608	21 - 36
614 - 686	38 - 49
686 - 698	50 - 51

Tabla 2.2: Bandas de frecuencias a ser utilizadas en TDT

- Banda 174 – 216 MHz: se utiliza para el servicio de radiodifusión con emisiones de televisión (canales de televisión 7 al 13). EQA.35

- Bandas 470 – 482 MHz y 686 – 698 MHz: están siendo analizadas y despejadas de acuerdo a los procedimientos establecidos en la leyes y reglamentos, para el servicio de radiodifusión de TDT.
- Banda 512 - 608 MHz: se utiliza para el servicio de radiodifusión con emisiones de televisión (canales de televisión 21 al 36). EQA.65
- Banda 614 - 686 MHz: se utiliza para el servicio de radiodifusión con emisiones de televisión (canales de televisión 38 al 49). EQA.70
- Banda 686 – 806 MHz: operan sistemas de televisión codificada terrestre (canales de televisión 50 al 69). EQA.75

2.3.2. Apagón Analógico en Ecuador

Para el año 2018 se espera que todo el país tenga únicamente la señal de televisión digital, este cambio se hará de manera progresiva comenzando desde el 2016. Para finales del 2016 se iniciará el apagón de las emisiones de televisión analógica en las principales ciudades del país como son Quito, Guayaquil y Cuenca; luego, para el 2017 se realizará el apagón en varias capitales de provincia; y, para el 2018 se apagará completamente la señal analógica en todo el país. [19]

2.4. Set Top Box (STB)

Un STB es un equipo encargado de recibir e interpretar la señal digital, ya sea por cable, satélite, o por IPTV. Por lo tanto, un STB hace que un televisor analógico pueda recibir la señal en formato digital. Los STB para televisión digital terrestre, y los que soportan el estándar ISDB-Tb se clasifican en: *Broadcast* TV, *Enhanced* TV, y *Advanced Services* [3].

***Broadcast* TV:** Brindan servicios tradicionales de televisión; algunos incluyen sistemas *Pay Per View* (PPV) donde el abonado paga por los eventos individuales que desea ver siendo una modalidad de televisión por suscripción, además se tiene herramientas básicas de navegación y permiten servicios de interacción local [3].

***Enhanced* TV:** Incluyen una interfaz para un canal de retorno, para brindar servicios interactivos remotos como por ejemplo: comercio electrónico, VoD, navegadores para Internet. Para poder ofrecer estos servicios deben poseer mayor capacidad de procesamiento y memoria en comparación con los STB de tipo “Broadcast TV” [3].

***Advanced Services*:** Posee un canal de retorno de alta velocidad y una capacidad de

procesamiento cerca de diez veces superior a los STBs "Broadcast TV", anteriormente mencionados, además brindan gran variedad de servicios de Internet e interactividad. Por lo general, en este tipo de STB viene incluido un disco duro [3].

2.4.1. Componentes de Hardware del STB [2]

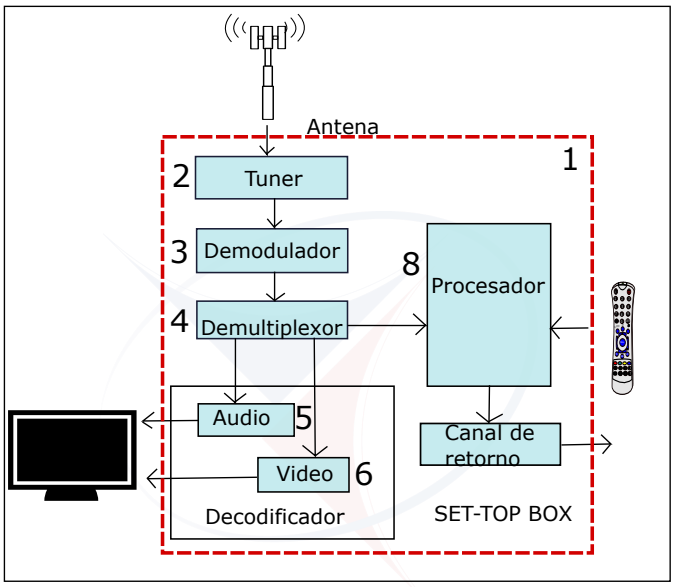


Figura 2.4: Esquema general de un STB, Fuente: [2, Fig. 1]

De acuerdo a la Figura 2.4 los componentes físicos que constituyen un STB son:

1. **Placa del sistema:** Es la placa madre o principal que se compone de un circuito impreso donde se conectan los circuitos integrados como el chipset, la memoria de acceso aleatorio (RAM), las ranuras de expansión y otros dispositivos.

2. **Sintonizador:** Es el componente que se encarga de captar y procesar la señal digital. El sintonizador selecciona un canal VHF o UHF (6 MHz en el caso de Ecuador), y convierte la señal de *Radiofrecuencia* (RF) a señal de banda base codificada, para que de esta manera sea más fácil de manipular por el resto de componentes. Este sintonizador puede ser una tarjeta de expansión, generalmente de tipo *Peripheral Component Interconnect* (PCI), o un dispositivo externo que se conecta al puerto USB.

3. **Modulador/demodulador:** El demodulador cumple con la función de muestrear la señal sintonizada y convertirla en bits a lo que se conoce como TS que contiene video, audio y datos codificados. El tipo de demodulador va a depender del estándar adoptado.

Una vez que se recupera el TS, se comprueba los errores, para reenviarlo al demultiplexor.

4. **Demultiplexor:** Se utiliza para separar la información proveniente del TS, realiza la extracción de los flujos elementales de audio, video y datos. Los paquetes de datos son identificados por un *Packet Id* (PID), para que el demultiplexor seleccione, demultiplexe y envíe paquetes específicos al decodificador correspondiente, por ejemplo, los paquetes con PID de video serán enviados al decodificador de video, al igual que los paquetes de audio y datos se envían al decodificador correspondiente.

5. **Decodificador de Audio:** El flujo de audio, proveniente del demultiplexor, se envía al decodificador de audio donde se descomprime para presentarlo en un formato analógico de audio (estéreo/mono) o digital.

6. **Decodificador de Video:** Descomprime los paquetes de video procedentes del demultiplexor y los convierte en una secuencia de imágenes para mostrarlas en formato en diferentes resoluciones de pantalla.

7. **Procesador gráfico:** Procesa las imágenes de acuerdo a la codificación utilizada. La *Graphics Proccessing Unit* (GPU) es un procesador especializado con funciones avanzadas de procesamiento de imágenes, en especial para gráficos 3D. Permite la interconexión entre los diversos componentes del STB, por ejemplo, el proceso de los comandos que utiliza el usuario a través de su interfaz, controlando el sintonizador y ajustando el dispositivo de conversión y codificación de señales de audio y video.

8. **CPU:** Procesador o cerebro de la STB. Sus funciones son: inicializar los diversos componentes de hardware del decodificador; supervisar y administrar el hardware; cargar datos e instrucciones de la memoria; y, ejecutar programas.

9. **Memoria:** Es responsable para el almacenamiento temporal de los datos entre el microprocesador y varios componentes de hardware, como pueden ser el motor gráfico, el decodificador de video, etc.

10. **Interfaces físicas:** Las interfaces más utilizadas para el canal de retorno son:
    Módems: Se utilizan para proporcionar al usuario servicios interactivos mediante un canal de retorno, donde se conecta el decodificador a un proveedor de servicio. Por lo general, están acoplados al fabricante del STB, pero también se puede instalar un módem externo.
    Y, **10 Base-T (Ethernet):** Interfaz utilizada para conectar el decodificador a una red, para compartir recursos y datos.

2.4.2. Componentes de Software de un STB [3]

Se presenta el modelo de capas de software de un STB, que se muestra en la Figura 2.5, y se describe a continuación.

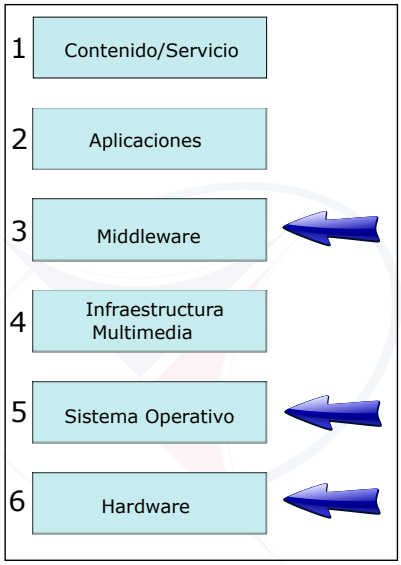


Figura 2.5: Arquitectura general de un STB

1. **Contenido/Servicio:** Es la capa que tiene los contenidos y servicios que son producidos en una transmisión de TDT. Por ejemplo: EPG, sistema PPV, juegos *on-line*, programas interactivos, etc.

2. **Aplicaciones:** Es la capa donde se encuentran las aplicaciones que son las que permiten acceder al tipo de servicio en la capa superior llamada Contenido/Servicio.

3. **Middleware:** En esta capa se realiza una interfaz entre el hardware del STB y el código de las aplicaciones. Gracias al *middleware* se puede acceder de manera transparente a las aplicaciones sin tener la preocupación de acceso al hardware. Un *middleware* para aplicaciones en TV digital consiste en máquinas de ejecución de los lenguajes ofrecidos, y bibliotecas de funciones, que permiten el desarrollo rápido y fácil de las aplicaciones [11].

4. **Multimedia:** Es la capa que contiene los componentes de codificación y decodificación, y módulos multimedia en general.

5. **Sistema Operativo:** El sistema operativo es responsable del funcionamiento del hardware, gestiona sus recursos y provee servicios a los programas de aplicación. Está organizado en capas, donde cada capa agrega una nueva funcionalidad. La capa principal de un SO de un STB es el kernel, la cual es almacenada en la memoria ROM. Cuando el STB se enciende, el kernel se carga en memoria virtual hasta que el dispositivo se apague; también es responsable de la gestión de los recursos de memoria, de las aplicaciones de tiempo real y de la transmisión de datos a alta velocidad. Además soporta multitarea, permitiendo ejecutar diferentes secciones de un programa y diferentes programas de forma simultánea. Todos los componentes de hardware dentro del STB requieren de un controlador o driver; programa que traduce órdenes provenientes del usuario a un formato que es reconocido por el hardware. Finalmente, el SO necesita incorporar un conjunto de APIs, para que los desarrolladores de software puedan escribir programas específicos de un entorno de SO de STB.

6. **Hardware:** Está constituido por elementos mencionados en la Sección 2.4.1.

2.4.3. Funcionamiento del STB [3]

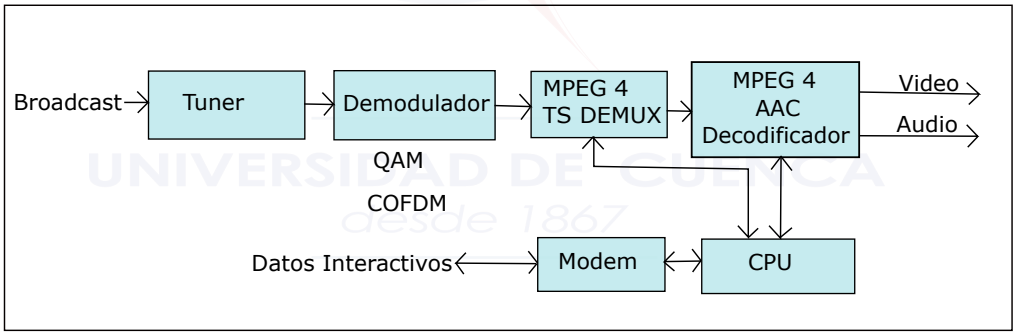


Figura 2.6: Esquema de funcionamiento de un STB

Como se muestra en la Figura 2.6, primero el STB selecciona la señal *broadcast* de TV sintonizando uno de los varios canales de entrada. Esta señal incluye información de audio, video y datos. La señal elegida se demodula en el canal de radiofrecuencia. Se debe tener en cuenta que tipo de señal se está transmitiendo, en este caso para ISDB-Tb se deberá implementar usando demodulación COFDM.

El resultado proveniente del demodulador es un TS de audio, video, y datos. El video y audio se lo detecta en H.264/AAC, y los datos en *Digital Video Broadcasting-Service Information* (DVB-SI). Una vez recibida la información de audio, video y datos, se tiene que separar

la información para tratar a cada señal de forma independiente, de esto se encarga el demultiplexor *Moving Pictures Experts Group* (MPEG) que selecciona y decodifica el audio y video del programa que el usuario desea ver.

Ahora la CPU tiene el control total de la operación y realiza la función de manipulación de datos específicos. Generalmente, la CPU hace uso de un sistema operativo de tiempo real en lo más alto de la capa de abstracción de hardware para el control de los recursos y procesos del STB.

Finalmente, el usuario podrá ver los contenidos. Los paquetes de audio y video son enviados al televisor una vez que la información fue decodificada, mientras que los datos aparecerán si es necesario o si los solicita el usuario. El STB puede contar con un canal de retorno para poder enviar datos a la cabecera, en TV abierta no existe el proceso de descifrar información.

2.4.4. Soluciones de Diseño de STB

El desarrollo y la migración hacia nuevas tecnologías, especialmente en el tema de televisión digital; está provocando un incremento en investigación e implementación de STBs, a continuación se describen las tecnologías *System-on-a-chip* (SoC) y *System in Package* (SIP).

2.4.4.1. System on a chip (SoC)

Las personas demandan cada vez más velocidad, y mejor experiencia con la conectividad, calidad y contenidos, todo esto con un sistema eficiente a bajo costo. Este sistema integra a todos componentes de un computador o cualquier otro sistema informático o electrónico, en este caso a los módulos del STB en un único circuito integrado o chip.

Arquitectura

La arquitectura de un SoC [25] está compuesta por los siguientes elementos:

- **Microprocesador:** Es el procesador, microcontrolador o núcleo *Digital Signal Processor* (DSP).
- **Controlador de memoria:** También conocido como *Memory Manager Unit* (MMU), es un circuito electrónico digital que se encarga de gestionar el flujo de datos entre el microprocesador y la memoria RAM.

- **Módulos de memoria:** Se encuentra la memoria RAM, ROM, EEPROM, y memoria Flash.
- **Chip gráfico:** También conocido como GPU, es el chip que se encarga de procesar los gráficos que se puede visualizar en la pantalla.
- **Buses:** Conjunto de conductores eléctricos en forma de pistas metálicas impresas sobre la placa base, por donde circulan las señales que corresponden a los datos binarios.
- **Componentes periféricos:** como generadores *Power-on Reset* (PoR), temporizadores o relojes a tiempo parcial y contadores-temporizadores.
- **Generadores de frecuencia fija:** Osciladores y/o lazos de seguimiento de fase o *Phase-Locked Loop* (PLL)'s.
- **Comunicación:** Wi-Fi, Bluetooth, y tecnología inalámbrica en general.
- **Interfaces externas:** USB, IEEE 1394/Firewire, Ethernet, USART o SPI.
- **Interfaces analógicas:** ADC's y DAC's.
- Reguladores de voltaje y circuitos de *Power Management*.

Ejemplos

En el mercado existe una amplia gama de modelos y fabricantes para estos sistemas, como por ejemplo: MIPS, ARM, AMD, *Broadcom*. Estos sistemas hacen que sea fácil para los fabricantes de equipos construir televisores y STBs, por ejemplo: Broadcom con su SoC BCM2835 y BCM2836 que corresponden al Raspberry Pi I y II respectivamente. Estos chips llevan integrados: CPU, GPU, DSP, SDRAM y puerto USB. Otro ejemplo BeagleBone con su SoC AM3358/9.

2.4.4.2. System in Package [4]

La tecnología SIP consiste en un número de circuitos integrados encerrados en un paquete. Los subsistemas como CPU, memoria, periféricos; se fabrican de manera independiente, son individuales para ser montados en el interior del paquete, mediante diversas técnicas como apilamiento vertical y apilamiento horizontal. Para sistemas de electrónica se pueden incluir múltiples dispositivos pasivos como filtros, cristales, condensadores, inductores, y resistencias, el espesor máximo del paquete de 1,10 a 1,60 mm. Mientras que los sistemas del pasado consistían en cajas voluminosas de cientos de componentes que realizaban una tarea, SIP consiste en un sistema de múltiples funciones como la computación, la comunicación, en un sistema de paquetes de tamaño pequeño y bajo costo.

SIP se presenta como una variación de *System on Package* (SOP). El SIP presenta estrictamente chips de silicio interconectados, por lo tanto está limitada por CMOS. La evolución

hacia la SOP, optimiza el rendimiento del IC. Junto con las funciones de RF y óptica embebida, aporta una sinergia a los sistemas en términos de coste, rendimiento y microminiaturización. El paradigma SOP es un paso para superar las deficiencias como en la integración de SoC y SIP, que están limitados por el procesamiento CMOS. Aunque la tecnología de silicio es ideal por las mejoras en la densidad de transistores de un año a otro, no es óptima para la integración de componentes de RF y ópticos.

2.4.4.3. Diferencias entre SoC y SIP

Un SoC tiene una única matriz de silicio y un proceso de fabricación único. En donde todos los subsistemas como son CPU, memoria, periféricos, OI; se basan en el mismo proceso y se colocan en un paquete. En cambio en el caso de SIP, los subsistemas son individuales y pueden ser fabricados de manera independiente; y se montan en el interior del paquete mediante diversas técnicas como apilamiento vertical o apilamiento horizontal.

2.4.4.4. STB Comerciales

Los diseñadores del receptor tienen la opción de elegir entre una familia de procesadores (CPU), procesadores gráficos (GPU) y sistema IP que proporcionan la mejor optimización del rendimiento, costo y potencia. Por lo que, combinando éstos módulos, se tiene como resultado sistemas embebidos, los cuales ya están en comercialización, en el Anexo A se muestran ejemplos de algunos STBs comerciales.

Capítulo 3

Trabajos realizados sobre diseño de STBs

Este capítulo presenta un resumen de algunos trabajos realizados acerca del diseño de los STBs, servirá como referencia frente a lo que se ha hecho y lo que falta por hacer en torno al desarrollo de estos dispositivos. A partir del estudio de varios documentos se determinarán datos relevantes, tendencias y perspectivas. De esta manera se generan nuevas preguntas y proyectos, y se amplía el conocimiento con el fin de aportar al estado del arte. Se ha separado por contenidos, donde en la primera sección indica trabajos relacionados al diseño de STB utilizando plataformas de bajo costo con respecto al hardware del STB; la segunda sección trata sistemas STB con soporte de Android; y finalmente, en la tercera sección se presenta trabajos con respecto a la integración de los STB, con el *middleware*.

3.1. Diseño de un STB utilizando plataformas de bajo costo.

Dentro de la arquitectura de un STB, mencionada en el Capítulo 2, intervienen varios elementos que permiten recibir correctamente la señal de TV digital, esta sección se enfocará en el estudio de varias alternativas acerca del diseño de un STB. Se mencionan algunos trabajos donde los autores integran hardware de bajo costo, así mismo el procedimiento para acoplar dichos elementos y la selección del sistema base para el correcto funcionamiento.

El trabajo [12] evalúa el potencial de la plataforma PandaBoard [1] para que trabaje como un STB basado en Android. El proyecto mencionado fue desarrollado en Suecia, donde el estándar de televisión digital es DVB-T2. El objetivo principal que plantearon los autores fue implementar un STB con la utilización de la plataforma Pandaboard con Pandroid[2], y un sintonizador USB.

Este proyecto [12] fue desarrollado en el 2011 donde el despliegue de tecnología del hardware y software para TV digital era limitado, los autores no pudieron habilitar el soporte para hardware dedicado para la decodificación de video y audio en Pandroid. En su lugar, los autores utilizaron la biblioteca FFMPEG, que es de código abierto y multiplataforma para el procesamiento multimedia.

Para la recepción de la señal digital utilizaron el *dongle* 290e PCTV Systems nanoStick T2[3], donde el procedimiento para el acoplamiento fue la reconfiguración y recompilación del kernel y software adicional con el fin de interactuar con Linux DVB-API que se utiliza para operaciones de sintonización y exploración de canales.

El cuestionamiento principal de este proyecto fue si el PandaBoard y el OMAP4[4] (procesador) en particular, eran lo suficientemente potentes en términos de rendimiento para ser utilizado como un STB. En la Figura 3.1, se muestra la adecuación de todo el sistema, además se tiene los resultados de las pruebas en la recepción de video, de acuerdo a la cantidad de FPS que obtuvieron, por ejemplo para SVT1 (SD) hubo un promedio de 70 FPS, ya que recupera toda la información porque el video se codifica con 29.6 FPS. SVT1 HD dio un promedio de 14,8 FPS, este valor es 50 % menor que el flujo de video a 29.6 FPS. En este caso no se mostraron todos los *frames* dando como resultado una experiencia de visualización pobre.

Aunque este trabajo no pudo ser concluyente en cuanto a la calidad en reproducción de video, los autores demostraron lo que puede lograrse con una plataforma de bajo costo con

[1] http://pandaboard.org/. PandaBoard: Plataforma de desarrollo basada en la Texas Instruments OMAP4430 sistema en un chip (SoC)
[2] http://pandaboard.org/content/pandroid. Pandroid: Ambiente Android para la plataforma PandaBoard
[3] http://www.pctvsystems.com. 290e PCTV Systems nanoStick T2: Sintonizador USB
[4] http://www.ti.com. OMAP4: Procesador de video desarrollado por Texas Instruments

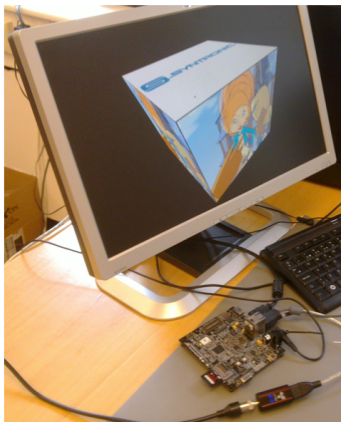


Figura 3.1: Adecuación del sistema, Fuente: [12, Fig. 4.8]

procesador ARM Cortex-A9 de doble núcleo junto con librerías FFMPEG[5] y OpenGL ES 2.0[6] en un sistema que ejecuta el sistema operativo Android, ésto es una buena aproximación que sirve como referente a este proyecto de tesis.

Por otra parte, los autores de [13] propusieron algo similar al trabajo anteriormente mencionado, su objetivo fue realizar un estudio de factibilidad para el desarrollo de un prototipo de STB a bajo costo, basado en sistemas Android. El STB se diseñó con un CPU allwinner A10 [7] como decodificador y un PCTV NanoStick como sintonizador y demodulador, este trabajo fue realizado en Indonesia donde el estándar de TV digital adoptado es DVB-T2.

Para la fase inicial requirieron un kit de desarrollo de STB (AllWinner A10 (Figura 3.2)), el cual utiliza Android. Para el sintonizador y demodulador se hizo uso del PCTV NanoStick. Prepararon el controlador del USB sintonizador usando Linux para ser combinado con allwinner A10. En la Figura 3.3 se muestra el sistema receptor de DVB-T2 de TDT, que consiste en una antena, decodificador, sintonizador, demodulador, y el monitor de televisión.

Los autores propusieron un STB donde utilizaron software Android para combinar con el hardware y ejecutar aplicaciones con funciones amigables para el usuario y de esta manera lograron buen rendimiento de recepción. Este trabajo demuestra la potencia computacional de una plataforma de hardware de bajo costo con la adecuada configuración y combinación con software.

[5] http://www.ffmpeg.org. FFMPEG: Es una librería para grabar, convertir y transmitir audio y video.
[6] http://www.khronos.org/opengles/. OpenGL ES 2.0: Es un estándar para las funciones de gráficos 2D y 3D en sistemas embebidos.
[7] http://www.allwinnertech.com/en/clq/processora/A10.html. winner A10: Dispositivo SoC diseñado por AllWinner Technology

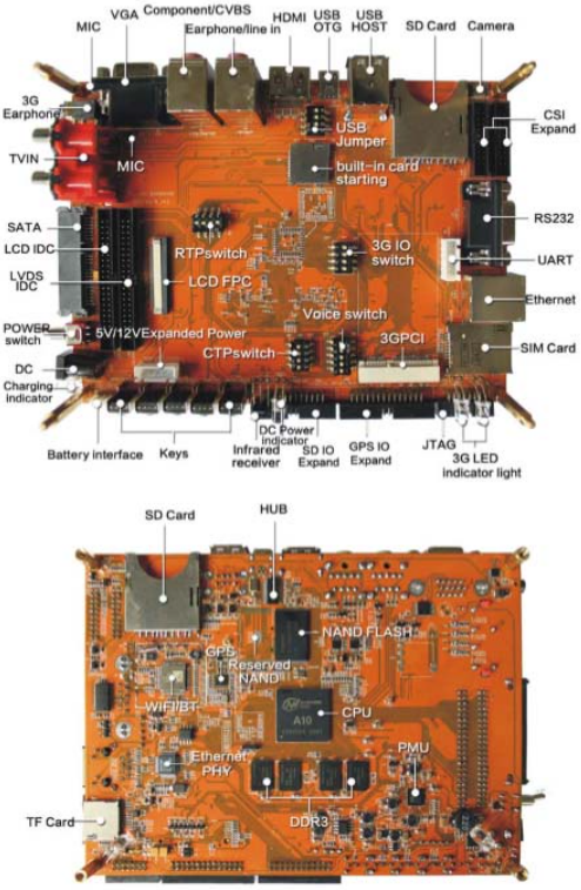


Figura 3.2: Allwinner A10, Fuente: [13, Fig. 3]

Figura 3.3: Sistema receptor de DVB-T2, Fuente: [13, Fig. 8]

3.2. Sistemas STB con soporte Android usando el sistema operativo Embebido Linux

Esta sección analizará el software necesario que dará apoyo a la plataforma de hardware que conforma un STB. Se estudia el sistema Android como tendencia por parte de los desarrolladores para obtener provecho de las aplicaciones. Y además, este estudio añade un enfoque a las funcionalidades adicionales de un STB.

En el trabajo [26] el objetivo de los autores fue adecuar un STB para que soporte Android bajo el sistema embebido Linux, al ser un sistema operativo de código abierto es posible aprovechar las aplicaciones si los decodificadores implementan un canal de retorno (conectado a Internet).

El proceso que se realizó consistió en la configuración del kernel Linux y *drivers* del dispositivo adicional para el apoyo de todo el componente de hardware. Posteriormente se instaló la máquina virtual Dalvik [1] en la parte superior del núcleo de Linux, y finalmente el soporte para toda la aplicación en el entorno STB.

Dentro del hardware consideraron los siguientes componentes: tarjeta DVB-S que fue apoyada por el sistema Embebido Linux, demodulador, controlador PCI, decodificador, placa madre Mini ITX, receptor de infrarrojos y la tarjeta LAN.

Una vez compilado el kernel, procedieron a trabajar con Dalvik Turbo[2] donde los desarrolladores de aplicaciones escriben programas en Java. Desarrollaron un sistema para dividir la pantalla del televisor en dos lados de visualización. En un lado, se puede trabajar con Internet y en el otro se puede ver programas de televisión.

Inicialmente, se cargó el sistema operativo Linux, embebido en una computadora personal y el dispositivo con la tarjeta sintonizadora de TV DVB-S; y comprobaron el funcionamiento y otras características de trabajo. En ésta se ejecutó una aplicación que consistía de dividir la pantalla para ver videos y navegar al mismo tiempo. Su principal contribución fue añadir soporte de aplicaciones de Android en el sistema nativo en Linux (sin usar la máquina virtual).

[1] http://www.dalvikvm.com/.Dalvik: Es una máquina virtual que se ejecuta en el sistema operativo Android.
[2] Dalvik Turbo: Es una alternativa para la aplicación de Google de la máquina virtual Dalvik. La máquina virtual se ejecuta la plataforma Java en dispositivos móviles compatibles.

3.3. Validación del Middleware con la utilización de una plataforma de bajo costo

Luego de haber realizado un estudio de los componentes de hardware necesarios de un STB en la Sección 2.4.1, es importante destacar algunos trabajos relacionados con la implantación del *middleware*, ya que éste proveerá el medio para la interacción del usuario mediante aplicaciones. Los trabajos que se describen a continuación servirán como referencia a este proyecto de tesis, debido a que se validan datos y errores de las pruebas realizadas en una plataforma que tiene integrado *middleware*, lo cual ayudará a mostrar la factibilidad del mismo.

En el trabajo [27], los autores propusieron una solución de bajo costo para validar un *middleware* utilizando una plataforma genérica como Raspberry Pi Modelo B, y una computadora convencional. Los autores encontraron en esta plataforma de bajo costo un potencial suficientemente interesante ya que incluye una API de código abierto, una herramienta para compilación cruzada y un simulador de PC, documentación y un foro en línea de apoyo técnico y discusiones. La plataforma ejecuta un sistema operativo basado en Linux. Esta elección no excluye el uso futuro de plataformas similares como son Beaglebone, Hackberry, los cuales son ordenadores de desarrollo de bajo costo.

El *middleware* fue compilado utilizando el compilador cruzado de Raspberry Pi, para luego ejecutar una aplicación NCL comparando con la versión de un computador convencional. Este estudio muestra que una placa de bajo costo es eficaz para desarrollar pruebas de concepto para la electrónica de consumo, pero requiere mejoras para ser un producto para el usuario final. La interfaz de usuario, objetos multimedia procesamiento y análisis de la aplicación tuvo un comportamiento similar a la PC.

En el proyecto [28] los autores propusieron realizar una incorporación del *middleware* Ginga en un STB basado en un SoC para el estándar Brasileño de TDT. Este diseño se implementó en una plataforma FPGA, que contiene un procesador Leon-3 y un decodificador de video H.264/AVC, que funciona a 200 MHz. Para entregar un sistema completo con Ginga utilizaron Linuxbuild[3] como sistema operativo, que proporciona los controladores de video y de propósito general de entrada/salida.

El objetivo principal fue generar un *middleware* en el SoC, imitando la implementación desarrollada en la PC de escritorio, para ésto se utilizó una configuración sugerida por los laboratorios Telemidia[4]. En este caso, portar, significó compilar sin errores, donde el objetivo principal fue satisfacer las dependencias. El *middleware* consiste en una serie de 18 paquetes donde el problema más común que encontraron fue la falta de un apoyo a las reglas de con-

[3] Linuxbuild: entorno de desarrollo de software de Aeroflex Gaisler para Linux
[4] http://www.telemidia.puc-rio.br/

figuración para compilación cruzada. En esta etapa, también se presentaron problemas con incompatibilidades de código, por ejemplo, el código específico para procesadores x86.

Un elemento crucial que consideraron es el motor gráfico, el *middleware* ofrece una variedad de opciones; el elegido para este trabajo fue DirectFB, que es una API gráfica diseñada para sistemas embebidos.

Utilizando el paquete de validación para GingaNCL [29], se realizaron pruebas en una máquina virtual que se comparó con el medio portado. Sin embargo, se observaron algunos problemas, en una aplicación no tuvieron buenos resultados, la aplicación consistía en que una prueba, muestra un azul por 3 segundos y luego cambia a un rojo por otros 3 segundos, poniendo fin a la aplicación. Lo que pasó en el medio portado es que la imagen azul tomó mucho más tiempo para mostrar, la transición a la imagen roja no se produjo y la aplicación terminó abruptamente.

Además de estas pruebas, utilizaron una rutina de evaluación del DirectFB[5]. Una rutina llamada `df andi` lograba sólo 1,4 FPS, se estableció un objetivo inicial de 30 FPS, por lo que es necesario una optimización de hardware. Los autores sugirieron que una solución para estos problemas sería implementar un hardware específico para gráficos y que sea transparente al *middleware*, además de un estudio más profundo sobre los recursos de la aplicación.

3.4. Conclusiones

Según los trabajos recopilados, es posible adecuar y trabajar sobre una plataforma de bajo costo para tener un STB, los trabajos han demostrado que se requiere un alto nivel computacional para implementar un STB. Además, la tendencia de los desarrolladores es utilizar el sistema operativo Android o el sistema operativo Linux, o la combinación de ambos mediante una máquina virtual, esto se hace para obtener beneficios utilizando aplicaciones con navegación web.

La mayoría de estas investigaciones han sido realizadas en Europa por lo que utilizaron el estándar DVB-T2, por lo que en este proyecto se tomará en consideración los procedimientos de los trabajos pero adaptando el estándar acogido en el Ecuador.

En el trabajo [27] los autores lograron implantar un *middleware* a una plataforma ARM y ejecutar aplicaciones por lo que esto muestra la factibilidad de poder implementar un STB completo en una plataforma como Raspberry Pi, Beaglebone, etc.

[5]http://elinux.org/DirectFB. DirectFB: Es una biblioteca de software para el sistema operativo GNU/Linux que proporciona aceleración gráfica de hardware.

El trabajo [28] presenta desafíos y soluciones de portabilidad del *middleware* Ginga en una plataforma SoC con estándar ISDB-Tb, al ser un sistema con recursos muy limitados se tuvo inconvenientes en la ejecución de aplicaciones, pero los autores lograron demostrar la factibilidad de portar Ginga en una plataforma de bajo costo. Entonces, este proyecto esta enmarcado a complementar algunos trabajos existentes enfocado al entorno Ecuatoriano.

Capítulo 4

Estudio de plataformas de hardware

En el presente capítulo se describen las plataformas de hardware más reconocidas en el mercado para desarrollo de proyectos en múltiples áreas. Estas plataformas son Arduino, Raspberry Pi y BeagleBone; en primera instancia se describen sus características de hardware y software, y sus usos. Además, se realiza una comparación para verificar la plataforma que mejor se adapta a este proyecto.

4.1. Arduino[1]

Arduino es una plataforma de código abierto, basada en software y hardware con lo cual se puede crear implementaciones fácilmente, con un entorno de programación simple y claro, con entradas y salidas analógicas y digitales para hacer más sencillo el uso de la electrónica para usuarios principiantes como usuarios avanzados.

Ivrea Interaction Design Institute creó Arduino en el año 2005 como un proyecto para estudiantes sin experiencia en electrónica y programación, como una herramienta sencilla para realizar proyectos, pero al llegar a un número de usuarios más amplio la placa se adaptó a nuevas necesidades y desafíos, de tal manera que permite a los usuarios crear prototipos basados en sus necesidades. Entre los proyectos basados en Arduino se tiene: sistemas domóticos, alcoholímetros, cubos LED, e incluso kits de análisis de ADN. Además Google lanzó un kit de desarrollo derivado de Arduino para teléfonos Android, para que éste pueda controlar luces, sensores, etc. conectados de Arduino. [14]

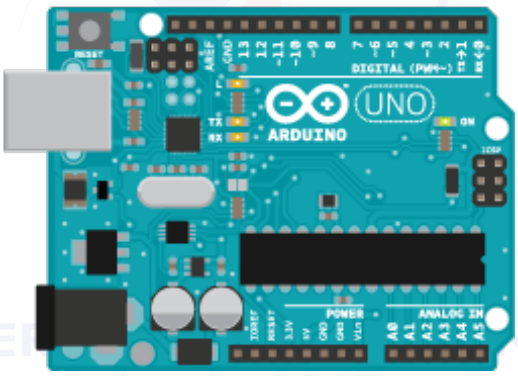


Figura 4.1: Arduino1, Fuente: [14]

4.1.1. Hardware [5]

El hardware consiste en una placa con un micro controlador principal Atmel AVR de 8 bits, encargado de realizar los procesos lógicos y matemáticos, además de controlar y gestionar los recursos de cada uno de los componentes externos conectados a la misma. Cuenta también con entradas de pines analógicos y digitales para adaptar fácilmente una amplia variedad de sensores eléctricos, que son controlados por el procesador principal junto con otros componentes como el Atmega168, Atmega328, Atmega1280 y el Atmega8, que son los más utilizados por su sencillez y costo.

[1] https://www.arduino.cc/

Además, Arduino cuenta con la ventaja de tener entre sus elementos principales puertos seriales de entrada/salida, lo que le permite conectarse por medio de un cable USB a una computadora para poder trabajar con ella desde nivel software, y controlar sus componentes.

Arduino Mega está basado en:

- Microcontrolador ATMega2560
- 54 pines de entradas/salidas digitales (14 de los cuales pueden ser utilizados como salidas PWM)
- 16 entradas analógicas
- 4 UARTs (puertos serial por hardware)
- Cristal oscilador de 16 Mhz
- Conexión USB
- Jack de alimentación
- Conector *In Circuit Serial Programming*(ICSP).
- Compatible con la mayoría de los *shields* diseñados para Arduino Duemilanove[2], diecimila o UNO
- No requiere *drivers* para Linux o Mac. Para la instalación de Arduino en Windows se necesita un archivo .INF del modelo.

4.1.2. Software

El software consiste en un entorno de desarrollo que implementa el lenguaje de programación *Wiring*[3] basado en la plataforma *Processing* y el cargador de arranque que es ejecutado en la placa. También puede pasar por alto el gestor de arranque y programar el microcontrolador a través del ICSP. Arduino está basado en el lenguaje C y soporta todas las funciones del estándar C y algunas de C++. Se programa en el entorno mencionado para que la placa controle cada uno de los componentes electrónicos conectados a ésta. Los programas se desarrollan usando el *Arduino Programming Language* (APL), luego de compilarse se cargan en la memoria del microcontrolador.

4.1.3. Ventajas

La plataforma Arduino presenta las siguientes ventajas:

[2]https://www.arduino.cc/en/Main/ArduinoBoardDuemilanove

[3]Wiring es un entorno de programación de entradas/salidas de código abierto para explorar las artes electrónicas, los medios materiales, la enseñanza y el aprendizaje de la programación informática y creación de prototipos con electrónica.

- Asequible
- Multiplataforma
- Entorno de programación simple y directo
- Software y hardware con opción de tener nuevas funcionalidades extendidas, permitiendo ampliar y añadir sus capacidades.

4.2. Raspberry Pi[4]

Es una plataforma de hardware de bajo costo, pequeña y potente que se conecta a un monitor de un ordenador o un televisor, utilizando un teclado y un ratón estándar; y, es de hardware abierto por lo que los esquemas necesarios para su fabricación se encuentran publicados para los usuarios que les interese utilizarlo y desarrollarlo, dichos esquemas se encuentran en la pagina oficial del Raspberry Pi[5].

Es capaz de realizar muchas de las tareas que hace una computadora de escritorio permitiendo a los usuarios conocer más de la computación y programación en lenguajes como Scratch y Python, ya que son lenguajes de programación flexibles y poderosos. Python es de alto nivel, es un lenguaje interpretado (permite escribir y ejecutar directamente un programa o *script* sin necesidad de un compilador). En Python no es necesario indicar si una variable es un número, una cadena de caracteres o un arreglo; el intérprete identifica el tipo de datos al ejecutar el *script*. Por otro lado, Scratch es una herramienta de programación visual que permite al usuario iniciar en el desarrollo de aplicaciones.

4.2.1. Hardware [6]

El Raspberry Pi original se basa en el SoC Broadcom BCM2835[6], que incluye un ARM1176JZF-S de 700 MHz de procesador, VideoCore GPU IV, y 256 MB de memoria RAM, aumentado en los modelos B y B+ a 512 MB. El sistema cuenta con MicroSD para medios de arranque y almacenamiento persistente.

A principios de febrero de 2015 fue liberado el Raspberry Pi 2 modelo B; que cuenta con un SoC Broadcom BCM2836, un CPU de cuatro núcleos ARM Cortex-A7 y un GPU de doble núcleo VideoCore IV; con un 1 GB de RAM con las demás especificaciones similares a las del modelo anterior como son 4 puertos USB, 40 pines *General Purpose Input/Output*(GPIO),

[4]https://www.raspberrypi.org/
[5]https://www.raspberrypi.org/
[6]https://www.raspberrypi.org/documentation/hardware/raspberrypi/bcm2835/. Broadcom BCM2835: El chip Broadcom usado en Raspberry Pi Modelo A, B, B + y el módulo de cómputo

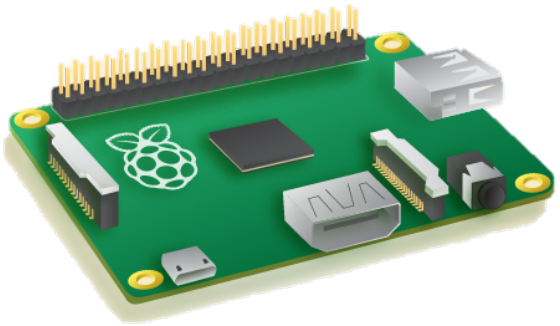


Figura 4.2: Raspberry Pi 1 Model A+, Fuente: [6]

puerto Full HDMI, puerto Ethernet, conector de audio de 3,5 mm combinado y video compuesto, interfaz de cámara (CSI), interfaz de pantalla (DSI), y ranura para tarjeta micro SD.

4.2.2. Software

El Raspberry Pi usa principalmente sistemas operativos basados en Linux. En Julio del 2012 se lanzó Raspbian, una distribución derivada de Debian optimizada para el hardware de Raspberry Pi. Por otro lado, existen distribuciones más específicas y ligeras como OpenELEC[7] y Raspbmc[8] (distribuciones con el centro multimedia XBMC).

Actualmente existen varias distribuciones de Ubuntu compatibles con Raspberry Pi 2. Se tiene una versión de Ubuntu con escritorio Lxde; basada en la versión de escritorio de Ubuntu 14.10. Y para el 2015 se desarrolló Ubuntu Mate en su versión 15.04 que tiene aceleración por hardware en video si se usa el reproductor omxplayer[9] y la distribución se basa en la versión del kernel de Linux 3.18. Además, se añadió esta distribución para que pueda ser instalada desde Noobs [10].

[7] http://openelec.tv/. OpenELEC: Es una distribución de Linux, basada en un centro de entretenimiento.
[8] https://osmc.tv/. Raspbmc: Actualmente OSMC principal centro de medios de comunicación se basa en el proyecto Kodi
[9] http://elinux.org/Omxplayer. omxplayer: Reproductor de video específicamente para la GPU del Raspberry Pi
[10] Noobs es el instalador oficial de sistemas operativos para Raspberry Pi

4.2.3. Ventajas

La plataforma Raspberry Pi presenta las siguientes ventajas:

- Computación de propósito general.
- Tamaño pequeño con todas las funciones de los ordenadores portátiles y de escritorio.
- Su procesador gráfico soporta 1080p.
- Capacidad de *overcloking* que permite obtener el máximo rendimiento del dispositivo, forzando la velocidad del procesador.
- Ya que el SO corre desde una tarjeta SD, puede cambiarse fácilmente con solo cambiar la tarjeta.

4.3. BeagleBone[11]

Es un miniordenador de bajo costo donde se puede ejecutar programas bajo un sistema operativo como Linux/Android 4.0, posee varias E/S y alta capacidad de procesamiento para el análisis en tiempo real proporcionado por un procesador de 720MHz ARM AM335x.

Existen dos modelos: BeagleBone original y BeagleBone Black. A continuación se detallan cada uno de estos:

4.3.1. BeagleBone (original) [7]

Utiliza SoC TI AM3358/9 basado en el procesador ARM Cortex-A8 que usa la arquitectura ARMv7-A. Se puede utilizar de manera independiente o como un USB o de expansión conectada en Ethernet para un *BeagleBoard* o cualquier otro sistema. Y cuenta con las siguientes características:

- Hasta 720 MHz ARM Cortex-A8 AM3358/9
- 256 MB de RAM DDR2
- Ethernet 10/100 RJ45, IPv4 e IPv6 de redes
- Ranura para tarjeta MicroSD
- Precargado con Angstrom ARM Linux Distribution
- Un puerto USB 2.0
- USB Hub dual de tipo USB mini-A OTG 2.0

[11] http://beagleboard.org/bone

Figura 4.3: BeagleBone original, Fuente: [7]

- Expandible E/S: 2 I2C, 5 UART, SPI, CAN, 66 GPIO, 8 PWM, 8 ADC
- +5V DC del conector o puerto dispositivo USB
- Consumo de energía de 300-500mA en 5V
- Dos bloques de 46-pines 3.3-V
- Tamaño: 3,4 "× 2.1"(86.4mm x 53.3mm)

4.3.2. BEAGLEBONE Negro [8]

El 23 de abril de 2013, BeagleBoard anunció oficialmente BeagleBone Black a un precio de aproximadamente la mitad del BeagleBone original, además de mejoras de procesador, memoria y salida de audio/video. Y cuenta con las siguientes características a diferencia del original:

- 1 GHz ARM Cortex-A8 AM3359
- 512 MB de RAM DDR3
- Ranura MicroSD para datos adicionales de usuario o sistemas operativos.
- Un puerto USB 2.0
- Puerto dedicado mini-USB 2.0
- Nueva salida micro-HDMI de audio/video
- Bajo consumo de energía de 210 a 460 mA en 5V

4.3.3. Ventajas

La plataforma BeagleBone presenta las siguientes ventajas:

- Procesador potente en velocidad y capacidades Cortex-A8.
- Memoria interna para el sistema operativo de 2GB incluida.
- Botones de encendido y reset.

4.4. Comparación de las 3 plataformas descritas

Debido a que existen diferentes modelos de cada uno de los dispositivos descritos en la sección anterior para la comparación se va a elegir los siguientes: Arduido Mega por ser el modelo más popular y estar diseñado para los proyectos más complejos que el Arduino Uno que es la plataforma más extendida y la primera que salió al mercado. Raspberry Pi 2 Model B por ser el modelo más reciente que cuenta con mejores características que sus versiones anteriores. Y, BeagleBone Black que tiene características comunes al BeagleBone original pero es más actualizado y de menor costo.

Se va a analizar los tres modelos indicados para saber cuál se adapta mejor a este proyecto ya que cada una tiene sus fortalezas y debilidades, y una plataforma es mejor que otra para una determinada aplicación. En la Tabla 4.1 se muestra una comparación de las características respectivas.

.	Arduino Mega	Raspberry Pi 2 Model B	BeagleBone Black
CPU	16 MHz	900 Mhz 4 núcleos	1 GHz
GPU	-	GPU de doble núcleo VideoCore IV	SGX530 aceleradora 3D
RAM	-	1 Gb	512 Mb
Memoria Flash	256 KB (8 KB utiliza el gestor de arranque)	Micro SD	Micro SD
Audio	-	Jack, HDMI	HDMI
Video	-	Jack, HDMI	Micro HDMI
Ethernet	-	Ethernet 10/100	Ethernet 10/100
Pines I/O Digitales	54 (15 proporcionan salida PWM)	40 pines GPIO	69 GPIO
Pines I/O Analógicos	16	-	7
Multitarea	No	Si	Si
Puertos USB	-	4 puertos USB 2.0	1
Sistema operativo	-	Distribuciones de Linux, Android	Debian, Android, Ubuntu

Entorno de desarrollo integrado (IDE)	Arduino IDE	Scratch, IDLE, Eclipse, cualquiera con soporte Linux	Python, Scratch, Linux, Eclipse, Android ADK

Tabla 4.1: Tabla comparativa de plataformas

En primer lugar, en cuanto a la memoria RAM se puede observar que Raspberry Pi 2 supera a los otros dispositivos, en gran medida al Arduino Mega y el doble del BeagleBone Black lo que ya supone una ventaja para el Raspberry Pi 2. El CPU de BeagleBone Black y Raspberry Pi es similar y supera al Arduino Mega.

En cuanto al video, audio y Ethernet se observa que Arduino Mega no cuenta con estas características, se tiene que agregar un *shield* que cuente con cada una de estas particularidades para integrarlo; a diferencia de Raspberry Pi 2 y BeagleBone Black que si tienen integrada estas características, para audio y video HDMI, y Ethernet 10/100.

Tanto el Raspberry Pi 2 y BeagleBone Black se rigen bajo el sistema operativo Linux lo que permite que sean capaces de ejecutar múltiples programas al mismo tiempo, por lo que les brinda la característica de multitarea, que no lo tiene Arduino Mega que puede ejecutar un programa a la vez por ser un diseño simple con programación de bajo nivel.

Una característica que tiene BeagleBone Black y el Raspberry Pi 2 es que cargan su sistema operativo en una tarjeta de memoria flash. Lo que facilita la migración de dicho sistemas y montajes sobre diferentes tarjetas.

4.4.1. Tabla con requerimientos de un STB

A continuación se presenta una tabla comparativa de las tres plataformas de hardware, donde se ha dado una ponderación de acuerdo al peso de la característica para que cumplan con los requerimiento de un STB.

Ponderación:
0: No cuenta con la característica
1: Cuenta con la característica pero no cumple con el requerimiento mínimo especificado
2: Cumple con el requerimiento mínimo
3: Cumple, y sobrepasa la especificación mínima

.	Arduino Mega	Raspberry Pi 2 Model B	BeagleBone Black
CPU	2	2	2
GPU	0	2	2
Sintonizador Integrado	0	0	0
Codecs: MPEG-2, MPEG-4	0	2	1
Decodificación de audio MPEG-1/2/3, AC-3, AAC	0	2	1
Salida de audio/video	0	2	2
HDMI con soporte 1080p	0	2	2
Salida de componente HDMI	0	2	2
Memoria no volatil minimo 2 MB	1	3	3
Interfaces externas	1	2	2
USB	1	3	2
Receptor IR	2	2	2
Total	7	24	21

Tabla 4.2: Tabla comparativa de plataformas con requerimientos de un STB

De acuerdo a la Tabla 4.2 los resultados son: Raspberry Pi =24 puntos, Beaglebone black=21 puntos, Arduino=7 puntos. Esta tabla servirá para decidir que plataforma usar en este proyecto.

4.5. Conclusiones

Según las descripciones y la tabla comparativa que se muestra en la Sección 4.4, se recomienda Arduino para proyectos de electrónica ya que puede interactuar con una amplia variedad de sensores y *shields*, sin circuitería externa. Para las aplicaciones que se conectan a Internet, se recomienda el BeagleBone Black o el Raspberry Pi 2 ya que ambos dispositivos son ordenadores

Linux que incluyen interfaces Ethernet y USB, lo que les permite conectarse a la red.

Para las aplicaciones que utilizan una interfaz gráfica de usuario se recomienda el Raspberry Pi 2 porque cuenta con un procesador gráfico integrado Video Core, que es capaz de decodificar flujos de video de 1080p y renderiza OpenGL; además tiene un conector full HDMI. A diferencia del BeagleBone Black que tiene soporte para la construcción de gráficos, pero no soporta 1080p, por lo que no es muy potente y aumenta el consumo de energía debido a este procesamiento.

El Raspberry Pi 2 cuenta con un procesador de 4 núcleos Broadcom BCM2836 ARMB7 y funciona a 900 MHz con 1 Gb de RAM lo que lo hace más robusto que el BeagleBone Black y además tiene apoyo de hasta 4 USB 2.0.

En conclusión, la elección depende de cuál va a ser el propósito de uso de la plataforma por eso en este proyecto se ha elegido trabajar con el Raspberry Pi 2, al mostrarse en la Tabla 4.2 tiene mayores ventajas que los demás dispositivos, y se va a adaptar mejor a la implementación que se va a realizar, ya que se necesita de un alto nivel de procesamiento gráfico, además de darle al usuario mayor número de periféricos y se cuenta con un amplio soporte de la comunidad Raspberry Pi.

Capítulo 5

Análisis de la plataforma a utilizar

En este capítulo se detalla la plataforma de hardware a implementar, justificando la elección de acuerdo a la comparación descrita en el Capítulo 4, así como el análisis de elementos de hardware adicional que se van a acoplar, el sistema operativo y *drivers* que complementarán el sistema. Se describe los pasos que se realizaron, así como los problemas, consideraciones y resultados en el proceso de implementación de un STB.

5.1. Plataforma Seleccionada

Se plantearon varias opciones de plataformas de hardware en la sección anterior (Capítulo 4) para construir un STB. El objetivo es utilizar componentes de bajo costo y, que se ajusten a requerimientos tales como: un alto poder computacional, y la manera que se adapta a la arquitectura general de un STB. Cada una de las plataformas presentadas tiene sus pros y sus contras para una determinada aplicación que no necesariamente será la misma.

El Raspberry Pi es una computadora pequeña que se rige bajo el sistema operativo Linux, por lo que las funcionalidades que ésta presta son amplias. Con las especificaciones descritas en la Sección 4.2, hace posible cualquier tipo de desarrollo. Esta placa se adapta mejor para aplicaciones que requieran interfaz gráfica, que fue la característica más sobresaliente (Ver Tabla 4.1). El Raspberry Pi, es una placa optimizada que permite tener un centro de entretenimiento (*media center*), un centro de descargas, un servidor, una nube propia y otras aplicaciones, a un bajo costo.

Para el presente trabajo se propone el uso de una placa de desarrollo con capacidades avanzadas, como es el caso del Raspberry Pi 2. Retomando las características que requiere un STB, que se mencionó en el Capítulo 2 y 4, el Raspberry Pi tiene lo siguiente, como se muestra en la Tabla 5.1.

Características de STB	**Cumple el Raspberry Pi**
Placa del sistema	✓
CPU	✓
GPU	✓
Sintonizador	✗
Demodulador	✗
Decodificador de audio	✓
Decodificador de video	✓
Memoria	✓
Interfaces físicas (USB, Ethernet)	✓
Salida de video HDMI con soporte 1080p	✓
Salida de Audio HDMI	✓
Receptor Infrarrojo	✗

Tabla 5.1: Tabla de características que cumple el Raspberry Pi

Por lo tanto el Raspberry Pi 2 es la plataforma que más se acerca a las características que debe tener un STB.

5.2. Análisis del sistema operativo a usar

Entre las opciones de sistemas operativos que se pueden implantar en el Raspberry Pi 2 se mencionan las siguientes: Raspbian[1], Raspbmc[2] y Ubuntu Mate[3]. Además hay un incremento en el desarrollo de otros sistemas operativos, como: Ubuntu Snappy Core[4], RasPBX[5], etc. En este proyecto se llevó a cabo la instalación y configuración de los sistemas operativos: Raspbian, Raspbmc y Ubuntu Mate.

5.2.1. Instalación y configuración de los sistemas operativos en Raspberry Pi

El sistema operativo, es el sistema base para la implantación del STB, en este caso se lo requiere para el respaldo y ejecución de procesos como: la instalación de programas, utilidades, librerías, herramientas, etc. que son necesarias para la adecuación del sistema. Esta implementación se la trabajó en dos etapas, la primera sobre la instalación y configuración del sintonizador, y la segunda sobre la implantación del *middleware*. Para lo cual el sistema operativo que se determine usar será muy importante a la hora de la conjunción de estas dos funciones.

A continuación se presenta un resumen del procedimiento de instalación del USB sintonizador, en el Anexo B.1 se encuentra el detalle:

1. Preparar las cabeceras (*linux headers*): se usan generalmente para realizar instalaciones referentes a los módulos del kernel y compilar los *drivers* gráficos para los chipsets.
2. Instalar Media Build (V4L): el proyecto LinuxTV acoge la última serie de módulos de los controladores del kernel de Linux para dispositivos V4L-DVB, para el uso de video analógico y fuentes de TV digitales en Linux.
3. Instalar w_scan: utilidad de línea de comandos que se utiliza para llevar a cabo exploraciones de frecuencias para transmisiones DVB. Es capaz de crear directamente archivos con los datos de la exploración. Posteriormente en la Tabla 5.3 se presentan varias opciones a este comando.

[1] https://www.raspbian.org/
[2] https://osmc.tv/. Raspbmc: Actualmente OSMC
[3] https://ubuntu-mate.org/
[4] https://developer.ubuntu.com/en/snappy/
[5] http://www.raspberry-asterisk.org/

4. Usar el comando w_scan para la sintonización y creación de listas de canales: mediante consola, una vez sintonizados canales, la información se guarda en una lista .xspf para ser ejecutada en el reproductor VLC, Kaffeine es otra opción a este reproductor.
5. Comprobar visualmente la calidad de reproducción: tanto el video y el sonido deben ofrecer buena calidad en HD, SD y *one-seg*, por lo que se comprobó visualmente usando los reproductores.
6. Configurar la decodificación por aceleración de hardware para mejorar la calidad de reproducción, debido a las interrupciones de imagen y sonido en la reproducción de canales SD y *one-seg* y al no tener ningún resultado en canales HD, se realizó la decodificación acelerada por hardware para aprovechar al máximo la GPU. Los pasos se detallan en el Anexo B.3.

En la segunda etapa se realizó la compilación del *middleware*, que requiere de un sistema robusto y compatible para desarrollos de programación, en futuros proyectos se ampliará con la creación de aplicaciones interactivas para obtener más beneficios del STB. Luego de trabajar con estos procesos en los tres sistemas operativos, se determinaron las características de cada uno de ellos, las cuales se describen en la Tabla 5.2.

N°	Paso	**Raspbian**	**Raspbmc**	**Ubuntu Mate**
1	Preparar cabeceras	Manualmente	Manualmente	Directo[6]
2	Soporte v4l	Si	Si	Si
3	Utilidad (*w_scan*)	Si	Si	Si
4	Sintonización	No se completó el paso 3	Si	Si
5	Reproducción	No	Si	Si
6	Middleware	Si	No	Si

Tabla 5.2: Tabla comparativa de SOs

Al realizar la instalación de las cabeceras, el proceso puede llegar a ser demorado, en Raspbian y Raspbmc no existe versión del Linux-headers para el kernel instalado por defecto (versión. 3.18). Por lo que es necesario construir manualmente el kernel como se describe en el Anexo B.2, mientras que en el caso de Ubuntu Mate se lo puede construir directamente, ya que existe una versión del linux-headers para esta version de kernel. El soporte de librerías, v4l, y utilidades se lo puede realizar en los tres sistemas operativos satisfactoriamente.

[6]Usando el sistema de instalación de paquetes de la distribución

De acuerdo a este análisis, al ser Raspbmc desarrollado como OSMC[7] (entorno dedicado a la multimedia) no se lo explotará como es debido, porque se quiere un entorno completo con *middleware* incluido y un entorno de sintonización y reproducción. Por otro lado, en Raspbian no se logró la sintonización ni reproducción de ningún canal de TV digital, mientras que en Ubuntu Mate se completó todo el procedimiento descrito.

5.2.2. Selección del sistema operativo

De acuerdo al análisis comparativo de los tres sistemas operativos para Raspberry Pi descrito en la Sección 5.2.1 Ubuntu Mate fue el que mayor facilidad para realizar el proceso de instalación. Fue creado por Rohith Madhavan y Martin Wimpress. MATE es un ambiente de escritorio derivado del código base de GNOME 2. Según el sitio oficial [30]: Ubuntu MATE es un sistema operativo estable, fácil de usar, con un entorno de escritorio configurable; se ajusta para personas que quieren el máximo rendimiento de sus equipos de escritorio, portátiles y *netbooks*. Para este proyecto, presenta un entorno ideal para la implementación de un STB.

5.3. Incorporación del Hardware

Según lo descrito en la Sección 5.1 el Raspberry Pi cumple con algunos de los requerimientos para ser un STB, tiene módulos necesarios de hardware y acoplamiento de software, inclusive puede prestar funcionalidades como: grabador de video digital con transmisión multimedia, y programación multitarea. Sin embargo, en el diagrama de bloques descrito en el Capítulo 2 sobre la arquitectura de un STB, el Raspberry Pi carece de hardware que realice el proceso de sintonización y demodulación de la señal de TV digital.

Entre las interfaces externas del dispositivo se encuentran los terminales GPIO por lo que es posible usarlos para una conexión directa con una antena y un sintonizador, y hacer el proceso de demodulación de la señal digital por software. Por otra parte, los puertos USB permiten conectar un receptor ISDB-Tb. En este proyecto se va a hacer uso de los puertos USB con el acople de un sintonizador de TV digital. Se ha decidido incorporar y configurar un sintonizador ISDB-Tb para que realice la etapa mencionada.

[7] https://osmc.tv/

5.3.1. Descripción del Hardware de recepción de ISDB-Tb

En esta investigación se realizó el estudio de dos módulos sintonizadores de diferentes fabricantes, como es el RTL2832U[8] Y SIANO MOBILE SILICON SMS4470[9], que se describen a continuación.

5.3.1.1. RTL2832U

El dispositivo RTL2832U es de la familia Realtek, son usados para radio y televisión digital, y ampliamente utilizados para simulaciones de radio definida por software (SDR). Este cuenta con dos elementos principales, el demodulador RTL2832U, y el sintonizador E4000E[10].

Figura 5.1: Elementos de RTL2832U, Fuente: [15, Fig. 3.3]

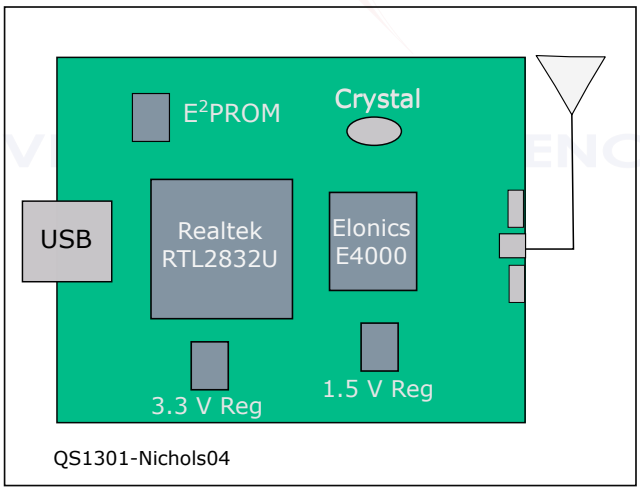


Figura 5.2: Arquitectura de RTL2832U, Fuente: [16]

El RTL2832U tiene un ADC de 8-bit, y puede funcionar a 3.2 MS/s. El rango de frecuencia es altamente dependiente del sintonizador utilizado, Elonics E4000 para ISDB-Tb ofrece un

[8] http://rfelektronik.se/manuals/Datasheets/RTL2832U.pdf
[9] http://www.electronicsdatasheets.com/pdf-datasheets/siano-mobile-silicon/sms4470/
[10] http://www.superkuh.com/gnuradio/Elonics-E4000-Low-Power-CMOS-Multi-Band-Tunner-Datasheet.pdf

rango de 470–862 MHz. Funciona tanto para recibir señales DVB-T y FM [31]. Las principales características de este dispositivo se las puede encontrar en [32].

5.3.1.2. Siano Mobile Silicon

Se trata de un módulo que se conecta mediante USB, se basa en un solo chipset modelo SMS4470. Esta placa tiene incorporada una antena que le permite detectar las señales RF. Mediante la antena y un sintonizador se recibe las señales que estén dentro de la cobertura de TV digital, soporta UHF/VHF/LOW-VHF(54MHz–806MHz) y *L-Band* (1.5GHz).

La salida es una señal demodulada seleccionada de un flujo de canales (*channel stream*). Los datos se introducen en un procesador de aplicaciones que es capaz de realizar la descompresión de video y audio, y su representación. Los chips receptores: sintonizador y demodulador de Siano se complementan con un paquete de software Host-API que incluye controladores y bibliotecas para los procesadores y sistemas operativos. Las especificaciones técnicas se detallan en [17].

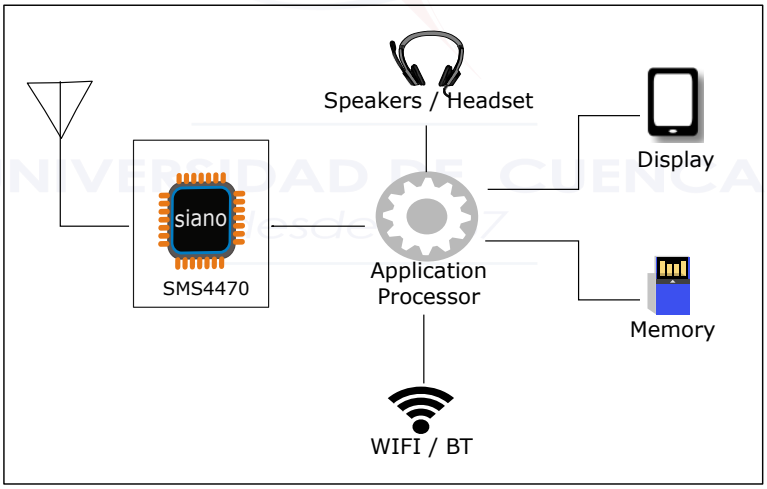


Figura 5.3: Esquema general Siano Mobile, Fuente: [17]

Para la implantación de este proyecto de tesis, se eligió la segunda opción, ya que el RTL2832U solo ofreció la posibilidad de receptar canales *one-seg* (Sección 2.1), y en este proyecto se requiere demostrar la factibilidad de implementación de un STB completo. Por lo que a continuación se detalla la instalación y configuración del USB Siano Mobile Silicon.

5.3.2. Instalación del dispositivo Siano Mobile Silicon

Los pasos se los detalla en el Anexo B.1. En primer lugar, se debe insertar el USB en uno de los puertos del Raspberry Pi, para comprobar que el equipo lo reconoce, mediante comandos como: `lsusb` y `uname -r`.

El siguiente paso es instalar las librerías necesarias. El proyecto LinuxTV tiene la última serie de módulos de los controladores del kernel de Linux para dispositivos V4L-DVB, por lo cual se instala un paquete llamado: `media_build`, el cual maneja todos estos controladores, necesarios para el funcionamiento del sistema.

Una vez instalados estos paquetes se debe reiniciar el sistema, y buscar los archivos de firmware necesarios para el dispositivo, los cuales son: `dvb_nova_12mhz_b0.inp`, `dvb_rio.inp` e `isdbt_rio.inp`; que se deben copiar a la carpeta `/lib/firmware`. Y una vez completado este proceso, se realiza una comprobación en el sistema que todo esté instalado, mediante el comando: `dmesg` como se muestra en los Listados 5.1 y 5.2.

```
usuario@usuario-desktop:~$ dmesg | grep dvb
[    7.500731] smsdvb:smsdvb_hotplug: DVB interface registered.
```

Listado 5.1: Mensaje de diagnostico dvb

```
usuario@usuario-desktop:~$ dmesg | grep sms
[    0.000000] Kernel command line: dma.dmachans=0x7f35 bcm2708_fb.
    fbwidth=640 bcm2708_fb.fbheight=480 bcm2709.boardrev=0xa21041
    bcm2709.serial=0x390f070e smsc95xx.macaddr=B8:27:EB:0F:07:0E
    bcm2708_fb.fbswap=1 bcm2709.disk_led_gpio=47 bcm2709.
    disk_led_active_low=0 sdhci-bcm2708.emmc_clock_freq=250000000
    vc_mem.mem_base=0x3dc00000 vc_mem.mem_size=0x3f000000  dwc_otg.
    lpm_enable=0 console=tty1 root=/dev/mmcblk0p2 elevator=deadline
    rootwait
[    0.832290] usbcore: registered new interface driver smsc95xx
[    2.257494] smsc95xx v1.0.4
[    2.311994] smsc95xx 1-1.1:1.0 eth0: register 'smsc95xx' at usb-
    bcm2708_usb-1.1, smsc95xx USB 2.0 Ethernet, b8:27:eb:0f:07:0e
[    6.335905] smsc95xx 1-1.1:1.0 eth1: renamed from eth0
[    6.520366] smsusb:smsusb_probe: board id=18, interface number 0
[    7.219958] smsmdtv:smscore_init_ir: IR port has not been
    detected
[    7.219992] smsusb:smsusb_probe: Device initialized with return
```

```
    code 0
[    7.500731] smsdvb:smsdvb_hotplug: DVB interface registered.
[    7.506565] usbcore: registered new interface driver smsusb
[   18.528863] smsc95xx 1-1.1:1.0 eth1: hardware isn't capable of
    remote wakeup
[   20.160807] smsc95xx 1-1.1:1.0 eth1: link up, 100Mbps, full-
    duplex, lpa 0x45E1
```

Listado 5.2: Mensaje de diagnóstico SMS (Siano Mobile Silicon)

5.3.3. Sintonización de canales

Una vez que se tiene instalado y configurado el acceso al dispositivo y el kernel, es necesario contar con un programa adecuado para la búsqueda y creación de las listas de los canales de televisión de los cuales se tiene los siguientes: `dvbscan`, `dvbv5-scan`, `w_scan`, `scan`, `scan-s2`. En la Tabla 5.3 se presenta un resumen de las características de cada uno de ellos.

En este proyecto se eligió `w_scan`, ya que permite trabajar sin ningún archivo original que contenga las frecuencias y especificaciones de los canales, en su lugar el archivo se genera durante la ejecución. El archivo consta de una serie de parámetros, como: ancho de banda, FEC, Modulación, etc. Se instala `dvb-apps`, la cual es un conjunto de herramientas, entre las cuales se encuentra `w_scan`. Con `w_scan` se puede buscar canales para ISDB-Tb, DVB-C, DVB-S/S2, DVB-T y canales ATSC. Con las opciones que se detallan a continuación:

- `-f TYPE`
  - "a" = ATSC,
  - "c" = DVB-C,
  - "s" = DVB-S/S2,
  - "t" = DVB-T [por defecto]
- `-c COUNTRY_ID`: Argumento obligatorio para las exploraciones ATSC, por cable y terrestre. Especifica el país en el que se intenta buscar los canales con identificador de dos letras, por ejemplo, EC para Ecuador.
- `-s SATELLITE_ID`: Argumento obligatorio para las exploraciones de satélite. Especifica el satélite que desea escanear con identificador en mayúsculas, por ejemplo, S19E2 = 19,2 ° Este.
- `-A N`: especifica tipo de escaneo ATSC
  1. = terrestre [por defecto],
  2. = cable,

Caract. \ scan	scan	dvbscan	w_scan	scan-s2	dvbv5-scan
última version	linuxtv-dvb-apps-1.1.1	linuxtv-dvb-apps-1.1.1	usa ultima versión	-	v4l-utils-1.4.0
Soporta DTV	DVB-S, DVB-C (solo Europeo), DVB-T, ATSC (VSB y QAM)	DVB-S, DVB-C (Solo Europeo), DVB-T, ATSC (VSB y QAM)	DVB-S, DVB-S2, DVB-C (Solo Europeo), DVB-T, DVB/T2, ISDB-T ATSC (VSB y QAM), DMB-TH (China)	DVB-S, DVB-S2, DVB-C (Solo Europeo), DVB-T, ATSC (VSB y QAM)	DVB-S, DVB-S2, DVB-C (Solo Europeo), DVB-T, DVB-T2, ATSC (VSB y QAM), ISDB-T
Archivo inicial requerido	✓	✓	✗	✓	✓
Genera archivo	✗	✗	✓	✗	✗
Genera canales para zap/xine/mplayer/vlc	✓	✓	✓	✓	✓
Genera canales.con para VDR	✓	✓	✓	✓	✓(luego de version 1.4.0)
Genera canales para kaffeine	✗	✗	✓	✗	✗

Tabla 5.3: Resumen de las características de los comandos

3. = cable y terrestre.

- `-o N`: formato VDR channels.conf
  6. = VDR-1.2 .. VDR-1.6 [por defecto],
  7. = VDR-1.7 (para DVB-S2)
- `-X`, genera zap/czap/xine en lugar de VDR channels.conf.
- `-x`, genera la salida de datos de sintonización inicial (DVB) en lugar de escanear VDR channels.conf.
- **`-k`**, genera channels.dvb para kaffeine en lugar de VDR channels.conf.
- `-h`, mostrar ayuda.
- `-H`, muestra ayuda de expertos.
- `-a N`, usa el dispositivo /dev/dvb/adapterN/ [por defecto: auto detect]

En este proyecto se va a utilizar el comando de ejecución con las opciones siguientes:

```
w_scan -a0 -c EC
```

Donde `w_scan` es la utilidad de la dvb-apps, `-a0` escoge el dispositivo USB conectado, `-c EC` se refiere al país Ecuador y automáticamente reconoce entre que frecuencias debería realizar la búsqueda. En el Listado 5.3 se muestra el resultado obtenido:

```
tune to: QAM_AUTO f = 557143 kHz I999B6C999D999T999G999Y999 (1:1:1)
    (time: 01:54.242)
        service = Avalpa1: MPEG2 MHP (Avalpa)
        service = Avalpa2: MPEG2 MHEG5 (Avalpa)
        service = Avalpa3: MPEG2 HBBTV (Avalpa)
        service = Avalpa4: MPEG2 TXT (Avalpa)
        service = Avalpa5: H264 (Avalpa)
        service = Avalpa6: HD H264 (Avalpa)
tune to: QAM_AUTO f = 671143 kHz I999B6C999D999T999G999Y999
    (1854:1854:1854) (time: 02:09.030)
        service = Ecuador SD1 ((null))
        service = Ecuador SD2 ((null))
        service = Ecuador 1seg ((null))
(time: 02:24.031) dumping lists (9 services)
..
Avalpa1   MPEG2 MHP;Avalpa:557142:B6:T:27500:2064=2:2068@3
    :0:0:1:1:1:0
Avalpa2   MPEG2 MHEG5;Avalpa:557142:B6:T:27500:2064=2:2068@3
    :0:0:2:1:1:0
```

```
Avalpa3  MPEG2 HBBTV;Avalpa:557142:B6:T:27500:2064=2:2068@3
    :0:0:3:1:1:0
Avalpa4  MPEG2 TXT;Avalpa:557142:B6:T:27500:2064=2:2068@3
    :1978:0:4:1:1:0
Avalpa5  H264;Avalpa:557142:B6:T:27500:2065=27:2068@3:0:0:5:1:1:0
Avalpa6  HD H264;Avalpa:557142:B6:T:27500:2066=27:0;2069:0:0:6:1:1:0
Ecuador SD1;(null):671142:B6:T:27500:769+772=27:770=por@17
    :0:0:59328:1854:1854:0
Ecuador SD2;(null):671142:B6:T:27500:1025+1028=27:1026=por@17
    :0:0:59329:1854:1854:0
Ecuador 1seg;(null):671142:B6:T:27500:257+260=27:258@17
    :0:0:59352:1854:1854:0
Done, scan time: 02:24.031
```

Listado 5.3: Archivo de resultado del *w_scan*

En el Listado 5.3, se observa el proceso de `w_scan`, donde se obtuvo la lista de frecuencias de canales sintonizados. La aplicación busca dentro de todo el ancho de banda y genera una lista. A continuación se necesita tener dicha información en un archivo para ser enviado al reproductor kaffeine o vlc, se realizaron pruebas con ambos reproductores. Se especifica en primer lugar la reproducción de los canales de TV digital con el programa Kaffeine.

El programa se instala desde el gestor de paquetes o mediante consola con el comando: `sudo apt-get install Kaffeine`. Se tiene la opción de escanear los canales con el botón *start scan*, previamente se elige la región y el dispositivo; y, como resultado muestra los canales sintonizados en la pantalla de reproducción.

En la Figura 5.4 se observa la lista de canales recibidos, y en la Figura 5.5 se observa los canales con mala calidad de imagen y sonido. Para mejorar el sonido y video se editó un archivo de configuración de Kaffeine el cual se encuentra en la siguiente ruta: `~/.kde/share/apps/kaffeine/xine-config`. En este archivo se configura el *driver* del video por defecto (video.driver) usando en su lugar *xshm*, ya que *xshm* es un *driver* de salida para Kaffeine que no utiliza superposición de video, y se implementa en uso intensivo de gráficos, la selección de este *driver* se muestra en el Listado 5.4.

```
# video driver
# { auto vdpau aadxr3 dxr3 xv vaapi XDirectFB opengl opengl2
    raw xshm xxmc none sdl xvmc }, default: 0
video.driver:xshm
```

Listado 5.4: Archivo xine-config de kaffeine modificado

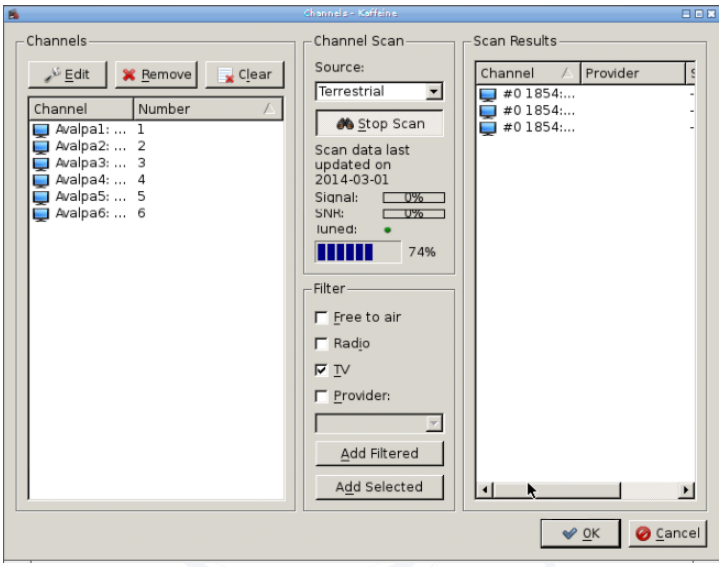


Figura 5.4: Prueba de sintonización en kaffeine

Luego de modificar el archivo se procede a realizar nuevamente la prueba de sintonización y reproducción de canales con el resultado que se muestra en la Figura 5.6. La imagen y sonido es más clara, pero no a una calidad aceptable, y al cambiar de canales la imagen se congela. Además no se logró reproducir en formato HD.

De igual manera se realizaron pruebas en el reproductor VLC, para lo cual se ejecutó el siguiente comando mediante consola: `w_scan -a0 -c EC -L > canales.xspf`. Donde se obtuvo un archivo canales.xspf con las frecuencias de TV digital sintonizadas. En el Listado C.1 se muestra una parte del archivo generado (el archivo completo se encuentra en el Anexo C).

```
        <title>0006. Avalpa6  HD H264</title>
        <location>dvb-t://frequency=557142857</location>
        <extension application="http://www.videolan.org/vlc/playlist
            /0">
                <vlc:option>dvb-bandwidth=6</vlc:option>
                <vlc:option>dvb-ts-id=1</vlc:option>
                <vlc:id>7</vlc:id>
                <vlc:option>program=6</vlc:option>
        </extension>
</track>
<track>
        <title>0007. Ecuador SD1</title>
```

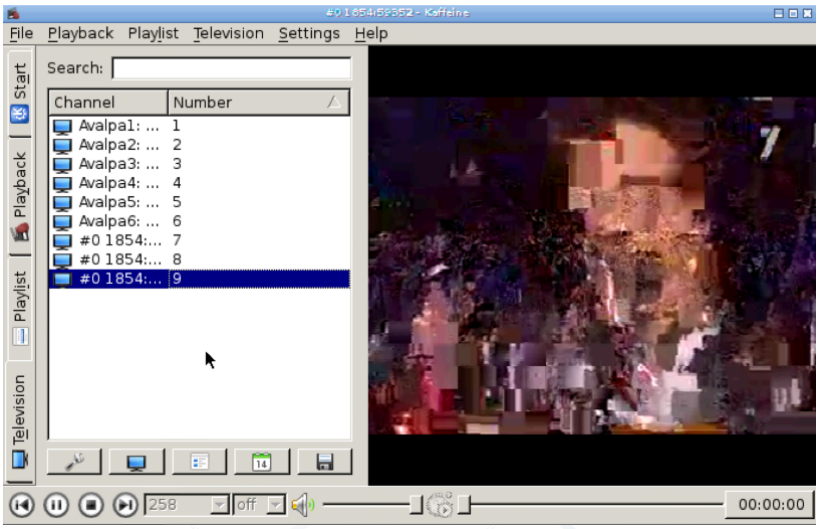


Figura 5.5: Prueba de reproducción de canales sintonizados

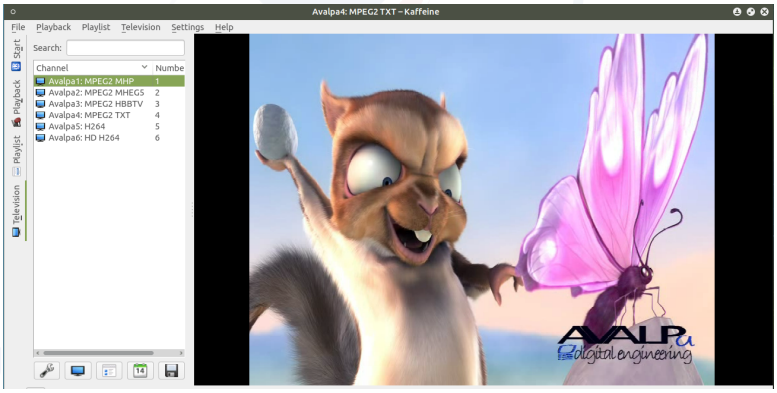


Figura 5.6: Reproducción de canal en kaffeine

```
        <location>dvb-t://frequency=671142857</location>
        <extension application="http://www.videolan.org/vlc/playlist
            /0">
                <vlc:option>dvb-bandwidth=6</vlc:option>
                <vlc:option>dvb-ts-id=1854</vlc:option>
                <vlc:id>8</vlc:id>
                <vlc:option>program=59328</vlc:option>
        </extension>
</track>
<track>
```

Listado 5.5: Archivo canales.xspf para VLC

Este archivo se abre en el reproductor VLC y se puede visualizar las transmisiones de los canales sintonizados, como se muestra en la Figura 5.7.

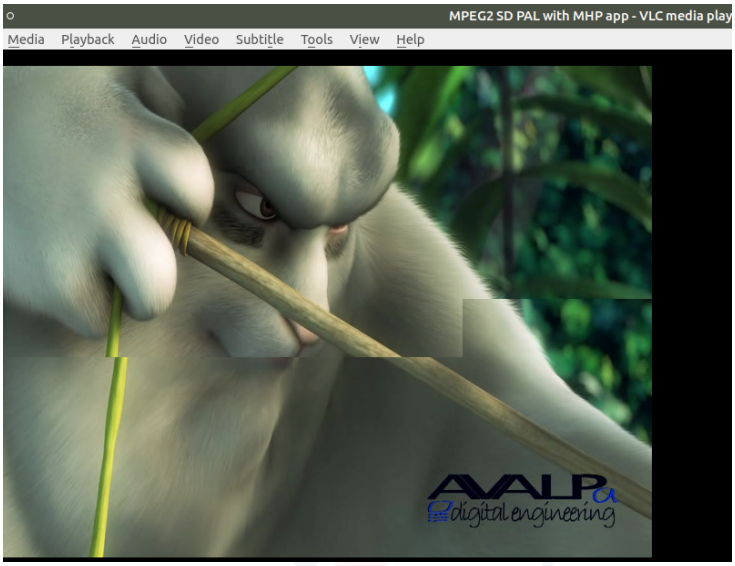


Figura 5.7: Reproducción de canal en VLC

Se obtuvieron interrupciones de imagen y sonido en la reproducción de los canales SD, HD y *one-seg*, además la imagen se congela al cambiar de canal, lo que evidencia el alto costo computacional de procesar la información que se recibe. La solución fue realizar una decodificación acelerada por hardware del reproductor VLC instalando los paquetes necesarios desde el repositorio de Ubuntu Mate. Es decir, se va a aprovechar todo el potencial de la GPU, estos pasos se describen en el Anexo B.3. Una vez realizado este proceso, se tiene como resultado un reproductor eficaz, se puede cambiar de canal sin problemas, y se recibe perfectamente la señal tanto en *one-seg*, SD y HD. El resultado se puede observar en la Figura 5.8.

Como resultado de las adecuaciones antes descritas para sintonizar y demodular correctamente las señales de TV digital. Se realizaron pruebas de transmisión de canales de TV abierta (Ecuador TV) y señales generadas en el laboratorio (Avalpa). Como resultado se tiene lo siguiente:

Canales recibido:
Avalpa-SD1: 25 FPS
Avalpa-SD2: 25 FPS
Avalpa-HD: 24 FPS
Ecuador TV-*one-seg*: 14.985 FPS
Ecuador TV-SD1: 29.97 FPS
Ecuador TV-SD2: 29.97 FPS

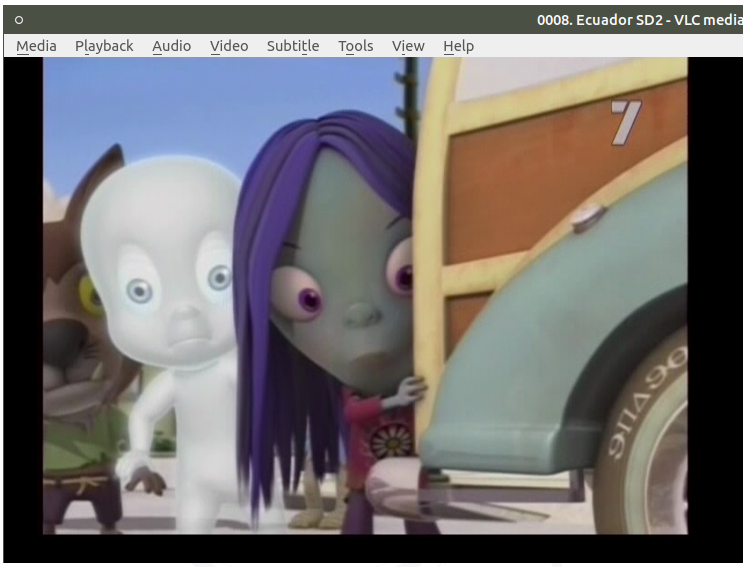


Figura 5.8: Reproducción de canal en VLC

5.4. Conclusiones

En este capítulo luego de seleccionar la plataforma de hardware sobre la cual se va a implantar el sistema completo del STB, se procedió a identificar el sistema operativo más adecuado, a partir de un conjunto de SOs. En los tres, se realizó el mismo procedimiento para la recepción de la señal y la implantación del *middleware*, donde al comparar los resultados obtenidos, se decidió usar el sistema operativo Ubuntu Mate, con el que se logró consolidar la arquitectura del STB. Para completar este sistema se analizaron dos dispositivos USB sintonizadores: Realtek2832U y SMS4470, resultando elegido el segundo pues brindó mejores características para este trabajo, por ejemplo, cumple con la sintonización en todas las bandas (*full-seg*), mientras que el primero permitía trabajar únicamente en *one-seg*. Dentro del proceso de adecuación del sistema se realizaron compilaciones de librerías, construcción de kernel, ajustes, e instalaciones de programas, además de la decodificación acelerada por hardware.

El Raspberry Pi por defecto decodifica el video por software, pero en su lugar se realizó la decodificación por hardware con lo que se obtuvo mejor visualización y sonido de las señales TDT. Además de ésto, se evaluó el sistema de acuerdo a la cantidad de fotogramas recibidos en todos los canales, donde se comprueba que se reciben los FPS necesarios, para el estándar ISDB-Tb.

Por otro lado, los receptores que implementan TV Digital interactiva deben tener una implementación del *middleware* Ginga, se realizó la instalación completa de éste donde se tuvo algunas consideraciones que se especifican en la siguiente sección. De acuerdo a los análisis presentados como los procedimientos, cuadros comparativos, instalaciones y especificaciones técnicas descritas en los anexos, se está en capacidad de ofrecer una referencia para que las personas que busquen implementar un sistema como éste tomen en consideración.

Capítulo 6

Implantación del Middleware

Este capítulo abarca el estudio de una capa más de la arquitectura del STB. Esta capa es conocida como *middleware*. Se dará una introducción, luego se verán los tipos, características, y ventajas de cada tipo. Finalmente, se tendrá los pasos de instalación y resultados de la investigación.

6.1. Introducción

Los STB están asociados a un software denominado *middleware*, que según la arquitectura del STB descrita en el Capítulo 3, se encuentra en una capa superior al Sistema Operativo. El significado viene de *middle* que se traduce como mitad haciendo referencia a una capa de software intermedia entre el hardware y el software del sistema, y se almacena en la memoria ROM como parte del *firmware*. El *middleware* forma parte de la arquitectura de un STB [3].

El *middleware* ofrece facilidades para que los contenidos y aplicaciones de TV Digital se puedan mostrar; permite a los desarrolladores de aplicaciones interactivas trabajar con STBs de distintos fabricantes ya que proporciona una capa de abstracción de software independiente del hardware. Además, consta de máquinas de ejecución de los lenguajes ofrecidos y librerías de funciones que permiten el desarrollo rápido y fácil de aplicaciones.

Se pueden crear tres tipos de aplicaciones para TV digital: las residentes que no requieren Internet, las recibidas por el canal de interactividad vía Internet y los programas no lineares; que se tratan de un programa de TV compuesto de video, audio y de otros datos principales transmitidos por el radiodifusor en conjunto con otros videos, sonidos, imágenes, textos, etc [33].

Cuando el usuario de TV Digital tiene acceso a Internet cuenta con interactividad que le permite navegar y obtener datos de Internet así como enviar datos en aplicaciones que lo requieran por ejemplo una votación.

6.2. Tipos de middleware

Existen diferentes tipos de *middleware* según el estándar (europeo, americano, japonés o brasilero); cada *middleware* cuenta con características específicas que siguen las recomendaciones de la norma GEM (UIT-T J.201)(UIT-T, 2001)[1] [34]. Para el estándar ISDB-Tb [35] [36], el *middleware* que utiliza es Ginga; nombre que nace en reconocimiento a la cultura, el arte y la continua lucha por la libertad y la igualdad del pueblo brasileño. La ITU-T ha publicado recomendaciones que son normas específicas de Ginga (ABNT NBR15606-2, NBR15606-5, NBR15606-7) [37].

Ginga fue definido por el SBTVD (Sistema Brasilero de TV Digital), el desarrollo de este *middleware* se obtuvo gracias a la investigación y cooperación de Laboratorio Telemidia de

[1] http://www.itu.int/itudoc/itu-t/aap/sg9aap/history/j201/j2101-es.html

la Universidad Católica de Río de Janeiro (PUC-Rio)[2] y por Lavid (UFPB)[3] Laboratorio de Aplicaciones de Video Digital de la Universidad Federal de Paraiba de Brasil. Y adoptado por el SATVD-T (Sistema Argentino de TV Digital Terrestre), a partir de la referencia de PUC-Rio el laboratorio LIFIA en la Universidad Nacional de La Plata desarrolló Ginga.ar [38].

6.3. Características de Ginga

Se divide en tres principales subsistemas integrados y se denominan:

- Ginga-NCL (para aplicaciones NCL declarativas)
- Ginga-J (para aplicaciones Java imperativas)
- Ginga-CC (Common Core, Núcleo Común)

La Figura 6.1 muestra la arquitectura de software para el *middleware* Ginga y se describe a continuación.

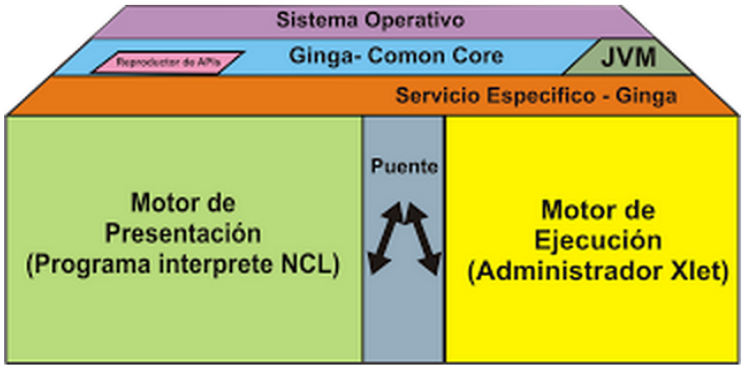


Figura 6.1: Arquitectura del *middleware* Ginga, Fuente: [18]

6.3.1. Ginga-NCL

Desarrollado por la Pontificia Universidad Católica de Río de Janeiro PUC-Rio, brinda una infraestructura de presentación para aplicaciones declarativas escritas en el lenguaje NCL [18]. La especificación de este subsistema se basa en las normas ABNT NBR 15606-2 [39] y ABNT NBR 15606-5 [39].

[2] http://www.puc-rio.br/index.html
[3] http://lavid.ufpb.br/

NCL

NCL [18] es un lenguaje declarativo XML basado en el modelo conceptual NCM, el cual proporciona facilidad para especificar aspectos de interactividad, sincronía en tiempo-espacio entre los objetos multimedia, adaptabilidad, soporte a dispositivos y producción en vivo de programas interactivos no lineares. Además usa el lenguaje de script Lua, basado en el paradigma imperativo, que es eficiente, rápido, ligero y diseñado para extender las aplicaciones.

Una aplicación puede ser generada o modificada en vivo ya que NCL posee una separación estricta entre el contenido y la estructura de un documento, permitiendo definir objetos de Ginga-NCL, debe ofrecer soporte a dos lenguajes procedurales, como son LUA y JAVA.

Lua

Lua [18] es un lenguaje imperativo, estructurado y diseñado para ser utilizado con otros lenguajes, permitiendo que una aplicación principal pueda ser ampliada o adaptada a través de scripts. Por lo que una de sus características es que posee una implementación ligera y extensible; también posee un recolector de basura (*Garbage-collection*) que sirve para no preocuparse por la asignación de memoria para los objetos, es decir un mecanismo de gestión de memoria. Lua se basa en el lenguaje C y Perl y puede registrar funciones C para que sean llamadas por el código Lua.

Entre sus características se tiene un alto rendimiento, bajo consumo de recursos, simplicidad, eficiencia, portabilidad, su licencia libre que combina con el escenario de la TV Digital. La portabilidad es importante cuando el *middleware* sea desarrollado para dispositivos con características contradictorias como la telefonía celular y STB.

6.3.2. Ginga-J

Procesa el contenido de los objetos Xlet[4] (aplicaciones tipo DVB-Java) por lo que Ginga-J representa el subsistema lógico de Ginga. Y un componente que tiene es el motor de ejecución de contenidos de procedimiento, compuesto por la máquina virtual de Java [18]. La especificación de este subsistema se basa en la norma ABNT NBR 15606-4 [39].

[4]Conjunto de clases que interactuan con un servicio de TV digital

6.3.3. Núcleo Común de Ginga (Comon-Core)

Reúne servicios para el Motor de Presentación (declarativo) y para el Motor de Ejecución (de procedimiento). Y representa la interfaz directa con el sistema operativo, accediendo al sintonizador de canales, sistema de archivos, terminal gráfico, entre otros [18]. El núcleo común de Ginga debe ser compatible con el modelo conceptual que se describe en la norma ABNT NBR 15606-1 [39].

6.3.4. Puente

Un puente formado entre los dos ambientes provee soporte a las aplicaciones híbridas con entidades especificadas en NCL, Lua y Java [40]. El puente bidireccional entre Ginga-NCL y Ginga-J se hace: i) en un sentido, a través de relaciones NCL soportados por Ginga-J, y ii) a través de *scripts* Lua que hacen referencia a métodos Ginga-J. En sentido contrario, a través de funciones Ginga-J que pueden monitorear y modificar cualquier evento NCL.

6.4. Implantación de Middleware en el Raspberry Pi

Los STB y televisores dentro del estándar ISDB-Tb que implementan TV Digital Interactiva deben tener una implementación del *middleware* Ginga. Actualmente, existen diferentes implementaciones de Ginga NCL cada una de las cuales tiene diferente nivel de completitud respecto a la norma.

En este proyecto se estudió Ginga NCL desarrollado en Brasil, y ginga.ar que corresponde a un Ginga NCL desarrollado en Argentina. No se instaló Ginga-Java, ya que se requería alto nivel de hardware para su correcto funcionamiento; a diferencia de Ginga NCL por lo que se encontró conveniente cargar este último para aprovechar de mejor manera el sistema. La implantación sobre la plataforma de hardware consistió en compilar librerías sin errores, para al final contar con un sistema embebido donde se pueda hacer pruebas de aplicaciones NCL.

6.4.1. Ginga Brasil

Ginga es el resultado del desarrollo de proyectos de investigación coordinados por los laboratorios Telemidia en la PUC-Rio y LAViD en la UFPB. Es una especificación abierta, de fácil aprendizaje y libre de royalties, permitiendo que todos produzcan contenido interactivo, lo que dará un nuevo impulso a las TVs comunitarias y a la producción de contenido por las grandes

emisoras. Extensiones de Ginga, sin embargo, se rigen por sus propias directivas. El entorno declarativo de Ginga, llamado Ginga-NCL tiene también una implementación de referencia en código abierto, desarrollada por el Laboratorio Telemidia de la PUC-Rio.

Adoptando la licencia GPLv2, el laboratorio Telemidia garantiza el acceso permanente a toda la evolución del código publicado en la Comunidad Ginga, sean cuales fueren sus aplicaciones y autores de aquí en adelante [41].

6.4.2. Ginga.ar

Ginga.ar es una implementación del estándar Ginga-NCL, desarrollada por el equipo de TV Digital del laboratorio LIFIA de la Universidad Nacional de La Plata (Argentina), a partir de la implementación de referencia Ginga-NCL creada por la PUC de Rio de Janeiro (Brasil)[5]. El proyecto desarrollado en Argentina intenta corregir y completar la implementación de Ginga NCL (PUCRio), y portarla a un STB [38]. El *middleware* Ginga.ar fue portado a los en STBs comerciales diseñados y producidos en Argentina. Estos son distribuidos en el proyecto de TV Digital del gobierno argentino. Ginga.ar corresponde a un *middleware* de software libre, las licencias utilizadas son GPLv2 y LGPLv2, e incluye librerías con otras licencias libres.

Argentina y Brasil mantienen posiciones diferentes en cuanto al *middleware* Ginga para TDT. En Argentina se impulsa el desarrollo de aplicaciones para Ginga-NCL, mientras que en Brasil hay una tendencia en incorporar GingaJ, que es software cerrado propiedad de Oracle.

6.4.3. Proceso de implantación del middleware en Raspberry Pi

En esta sección se describe el proceso para embeber el *middleware* Ginga NCL, en la plataforma de hardware de propósito general (Raspberry Pi). El siguiente procedimiento se hace tanto para Ginga NCL desarrollado en Brasil como para Ginga desarrollado en Argentina. Consiste en los siguientes pasos:

1. Instalar dependencias y prerrequisitos.
2. Obtener y compilar el código de Ginga.
3. Analizar errores, correcciones, parches.
4. Realizar pruebas de funcionamiento.
5. Determinar el sistema más estable entre los dos *middlewares*.

[5] http://tvd.lifia.info.unlp.edu.ar/ginga.ar

En el Anexo B.4 se detalla los pasos de instalación, errores y modificaciones de Ginga NCL brasileño, y en el Anexo B.5 se describe el caso de Ginga NCL argentino. Como se muestra en los anexos mencionados, Ginga.ar presentó mayor facilidad de instalación y compilación que Ginga de Brasil, que requirió la edición de algunos *scripts*, aplicación de parches, y cambios en el código para completar la compilación.

Por ejemplo para la instalación de `DirectFB-extra` que provee paquetes adicionales de fuentes y gráficos se debe modificar los archivos `Makefile.am` para enlazar objetos de bibliotecas adicionales; para `FusionSound` se debe modificar algunas lineas del archivo `ifusionsoundmusicprovider_ffmpeg.c` porque al intentar compilar se obtuvo errores al estar en una versión desactualizada; de igual manera para Lua se edita el archivo `Makefile` aumentando `-fPIC` en `CFLAGS`, esto es necesario para enlazar la biblioteca Lua a cualquier otra que quiera construir código independiente de la posición; además para la instalación de los paquetes de Ginga se deben realizar algunos cambios como: crear enlaces simbólicos, en `telemidia-links-cpp` el `Makefile` está configurado para una versión antigua de `DirectFB`, por lo que se crea un enlace para la carpeta `/include`; en `gingacc-cpp/gingacc-player` se crea un enlace simbólico al archivo `core.so`; y en `gingacc-cpp/gingacc-tuner` se debe incluir la biblioteca `unistd.h` en los archivos `SocketProvider.h` y `UnicastProvider.cpp`. Por último, para la instalación de Ginga Argentino se debe modificar el archivo `FindGlib.cmake` ya que está configurado para máquinas i386 y se requiere que compile para arm.

Una vez compilado Ginga se determina entre los dos *middleware* el sistema más estable mediante pruebas. Las siguientes aplicaciones se ejecutaron en el Raspberry Pi con tanto con Ginga Brasil como con Ginga Argentino embebidos en dicho sistema. La Figura 6.2 es el resultado de la aplicación: "El ahorcado"[6], que es un juego clásico, en el que el objetivo es adivinar una palabra o frase. Este juego fue desarrollado en NCL y se lo puede encontrar en la pagina oficial de Ginga.ar. Por otro lado, se tiene en la Figura 6.3 el resultado de la versión de "sokoban"[7], desarrollada en NCL y Lua, éste es un juego que consiste en acomodar las cajas en la ubicación correcta, utilizando para ésto las flechas del control remoto o el teclado.

Por otro lado, se realizaron un conjunto de pruebas NCL y Lua que validan la funcionalidad de una implementación de Ginga, que son independientes al desarrollador. Las pruebas se encuentran en la página oficial de Ginga Argentino[8], con la documentación para cada prueba que contiene una descripción del caso, una imagen de salida esperada, y una especificación de la funcionalidad evaluada. A continuación se presenta un resumen de las pruebas, que han sido agrupadas en áreas de funcionalidad:

- **Grupo *Layout*:** Pruebas de acuerdo a atributos de posicionamiento de imágenes.

[6]Descarga: http://tvd.lifia.info.unlp.edu.ar/ginga.ar/index.php/apps-menu
[7]Descarga: http://tvd.lifia.info.unlp.edu.ar/ginga.ar/index.php/apps-menu
[8]http://tvd.lifia.info.unlp.edu.ar/ginga.ar/index.php/test-suite-de-especificacion-ginga

Figura 6.2: Captura de aplicación Ahorcado

- **Grupo Descriptores:** Redefinición de atributos, tamaño y posición de imágenes.
- **Grupo Media (Text, imágenes, HTML, HTTP, Lua, Audio y Video):** Propiedades de texto, imágenes, páginas web, audio, video.
- **Grupo Filesystem (Estructura y acceso):** Acceso a distintas medias en diferentes niveles de la estructura de directorios.
- **Grupo Configuración (Ciclo de vida, Ambiente de ejecución):** Especifica el control del ciclo de vida de una aplicación, y cambio en la resolución (480p, 720p, 1080p).
- **Grupo de Comandos:** Actualización de texto.
- **Grupo Botones de control remoto:** Captura de teclas del control remoto.
- **Grupo Conectores:** Acciones con retardo (*delay*).
- **Grupo Canal de retorno:** Acciones entre server y cliente.
- **Grupo Contextos, Reglas y Switches:** Cambios de contexto mediante el uso de reglas y *switches*.
- **Grupo Animaciones:** Animaciones NCL.
- **Grupo Módulos Lua:** Propagación de eventos Lua.
- **Grupo BenchMarking:** Pruebas con un medio de medición externo (cronómetro).

Como se puede observar en los resultados descritos en la Figura 6.4. Se realizaron 114 pruebas, de las cuales 71 se ejecutaron correctamente, según las imágenes mostradas y esperadas, 26 no se lograron verificar ya que requerían de una interfaz con los botones del control remoto o interacción con el receptor. En 10 pruebas se necesitó de la sintonización de canales, debido a la falta de integración entre el vizualizador y el *middleware* Ginga; no se verificó estas opciones. Por otro lado 3 pruebas se produjeron con error, mientras que 4 no mostraron ninguna imagen,

Figura 6.3: Captura de aplicación Sokoban

ni errores. A continuación en la Tabla 6.1 se muestra dichos resultados desglosados para cada grupo.

Grupo	**Resultados / Pruebas**	**Descripción de los errores**
Layout	11/11	-
Descriptores	11/11	-
Media/Texto	5/5	-
Media/Imágenes	4/4	-
Media/HTML	2/4	Existió un problema de posicionamiento de una y múltiples páginas web
Media/HTTP	2/2	-
Media/Lua	4/6	No se probaron ya que se necesitaba los botones del control remoto
Media/Audio y Video	5/9	Un caso no probó ya que se necesitaba los botones del control remoto y tres requerían un video principal reproduciéndose conjuntamente con la aplicación
Filesystem Estructura y Acceso	1/5	Requerían interacción con el control remoto

Configuración/Ciclo de vida	5/5	-
Configuración/Ambiente de ejecución	3/3	-
Editing Comands	1/1	-
Botones de control remoto/Captura de Teclas	2/2	-
Botones de control remoto/Funciones numéricas	1/1	-
Botones de control remoto/Funciones interactivas	3/5	Requerían interacción con el control remoto
Botones de control remoto/Interacción con funcionalidad del receptor	0/5	Se requería la interacción con servicios como *Closed Caption*, *Second Audio Program*(SAP), menú, EPG, info, volumen, *mute* y *channel*, funciones que no se contemplaron en el STB
Conectores	0/2	Tiene acciones con *delay* donde los dos casos requerían del control remoto para su verificación
Canal de Retorno	0/1	No se realizó la conexión para el caso de prueba
Contexto, Reglas y *Switches*	2/3	Requerían interacción con el control remoto
Animaciones	0/2	Requerían interacción con el control remoto
Lua	8/9	Requería de la lectura y escritura dentro del carrusel de datos, función no habilitada
Lua - Eventos	1/4	Requerían los servicios del receptor como *Closed Caption*, SAP, menú, EPG, info, volumen, *mute* y *channel*
BenchMarking Inicio de Aplicación	0/6	Necesitaba la sintonización de un canal conjuntamente con la aplicación
Tiempo de respuesta	0/6	Necesitaba la sintonización de un canal conjuntamente con la aplicación
Carga y uso	0/2	Necesitaba la sintonización de un canal conjuntamente con la aplicación

Tabla 6.1: Tabla de resultado de las pruebas

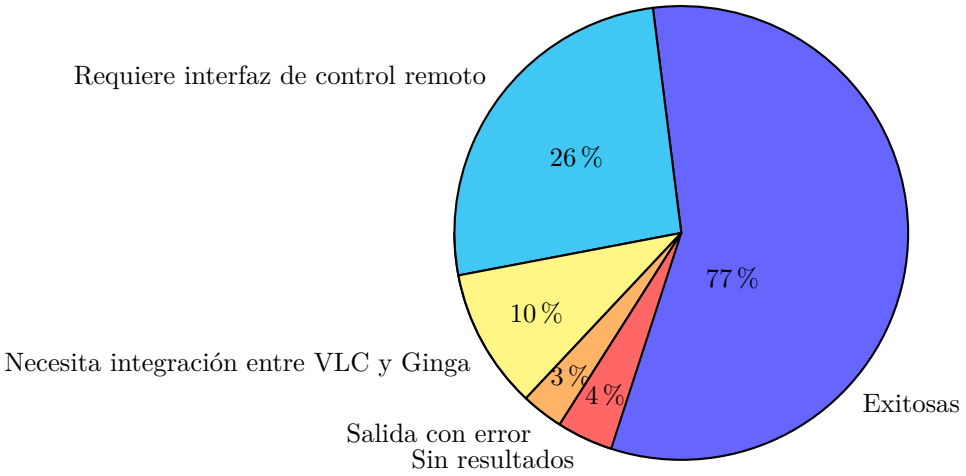


Figura 6.4: Resultados de las pruebas

6.5. Conclusiones

En este capítulo se presentaron retos para portar Ginga en el Raspberry Pi, problemas y soluciones en las configuraciones e instalaciones del *middleware*. En primer lugar se presentó un resumen de todo lo relacionado con el *middleware*, una vez acogidos estos conceptos, se procedió a la implantación sobre la plataforma de hardware, se comparó Ginga NCL desarrollado en Brasil y Ginga NCL desarrollado en Argentina, este último presentó mayores facilidades de compilación, así como mejores resultados en las pruebas; con respecto a éstas, se ejecutaron 114, de las cuales 71 se consideraron exitosas, de acuerdo a la salida y tiempo de ejecución esperado; sin embargo, las 43 pruebas restantes que no dieron los resultados esperados fue debido a que no se contempló el uso del control remoto y algunas funciones en el receptor. De acuerdo a los procesos y resultados, se decidió que para este proyecto se trabajaría con el *middleware* Ginga.ar.

Capítulo 7

Conclusiones y Recomendaciones

En este capítulo se describe las conclusiones que se obtuvieron en este proyecto de tesis, de acuerdo a los objetivos presentados en el Capítulo 1. Así mismo se dará algunas recomendaciones y sugerencias de trabajos futuros de acuerdo al desarrollo de esta investigación.

7.1. Conclusiones

Al inicio de este proyecto, se estudió los conceptos y temas relacionados con el área referente a este proyecto. A continuación se presentan las conclusiones en cada etapa del proyecto.

De acuerdo al primer objetivo planteado en el Capítulo 1 que consistió en recopilar trabajos relacionados con el diseño de STB, se estudiaron varias investigaciones que permitieron conocer el desarrollo de STBs en cuanto a plataformas de hardware, acople de dispositivos, e implantación de *middleware*. Estos trabajos demostraron la factibilidad de implementar un STB completo y dedicado a bajo costo, además de servir de base para tener una idea más objetiva sobre los posibles problemas y consideraciones que se deberían tomar en cuenta para realizar este proyecto.

A partir del estudio de los trabajos presentados en el Capitulo 3, se realizó una investigación sobre plataformas de hardware que podrían ser evaluadas para utilizar en este proyecto y con la recopilación de información en el Capítulo 4 se compararon las plataformas más populares del mercado. En el Capítulo 5 se presenta la plataforma de hardware escogida sobre la cual se desarrolló este trabajo, así como las opciones de dispositivos (sintonizadores) para la recepción de TV Digital, cumpliendo con el objetivo presentado de: proponer un procedimiento que permita evaluar diferentes dispositivos para la recepción de la TDT y para el procesamiento de la señal utilizando ordenadores de bajo costo, se analizó los USB sintonizadores RTL2832U y SMS4470 y se evaluaron mediante el correcto acople al sistema base formado por el Raspberry Pi y su sistema operativo, la sintonización y demodulación de la señal de TV digital. De las dos plataformas de sintonización analizadas, la mejor opción fue SMS4470 debido a que este dispositivo sintoniza bandas full-seg lo que permite trabajar con diferentes formatos de video como HD, SD y *one-seg*, por otra parte el sintonizador RTL2832U sólo permite trabajar en este último que es para dispositivos móviles.

En el Capitulo 5 se plantea y se implementa la integración entre el Raspberry Pi, y el USB receptor, con lo cual se obtuvo un sistema capaz de sintonizar canales de TV digital. La plataforma de hardware implementada debe ser lo suficientemente robusta como para reproducir la señal de TV digital de manera fluida y que permita al usuario contar con un sistema estable, sin embargo, al visualizar y escuchar los contenidos de diferentes canales no se obtuvo los resultados deseados ya que las imágenes y sonido se congelaban, y no se logró cambiar de canales sin que se colgara el sistema, para dar solución a este inconveniente se modificó las características por defecto del Raspberry Pi, mediante la edición de archivos de configuración de los reproductores, lo que mejoró la calidad, pero no significativamente. Para remediar este problema, se buscó otra opción, esta fue la decodificación acelerada por hardware del VLC para aprovechar al máximo las CPU y GPU.

Para la implementación del *middleware* sobre el Raspberry Pi, se analizaron opciones de *middleware* existentes, entre la opciones de desarrolladores se tomó en cuenta a Ginga NCL desarrollado en Brasil y el desarrollado en Argentina, y se decidió implementar ambos, para lo cual se tuvo que modificar códigos, librerías y ejecutar aplicaciones; lo que permitió evaluar la compatibilidad del sistema. Se obtuvo mejores resultados tanto en el proceso de compilación como en las pruebas realizadas con el Ginga NCL desarrollado en Argentina.

Acorde a los objetivos planteados se mostró la factibilidad de acoplar un STB implementado en una plataforma de hardware popular en el mercado, teniendo como resultado un prototipo como un paso previo a la implementación en un sistema profesional de bajo costo.

7.2. Recomendaciones

Al momento de la sintonización de canales, es recomendable asegurarse de la calidad de señal que se está queriendo recibir, ya que a veces no es problema del sintonizador si no del transmisor, o potencia de la señal.

Sería necesario conformar un grupo de trabajo para continuar con este proyecto dando una alternativa para la conjunción del *middleware* con el sintonizador y contar con un STB completo, aunque se logró por separado el funcionamiento de cada uno de ellos sería importante tener el VLC funcionando sobre Ginga; y tener la posibilidad de evaluarlo y mejorarlo.

Pese a que transcurre el tiempo y el apagón digital se acerca en nuestro país, y en las instituciones educativas existe aún un alto desconocimiento sobre el tema relacionado con TDT, se debería realizar una mayor difusión sobre el tema, y fomentar la investigación dentro de este campo por parte de las entidades competentes.

Es necesario, continuar con grupos dedicados al área de TDT, para desarrollos en cuanto a aplicaciones interactivas debido a la gran falta, necesidad y demanda que existe en el mercado. Así mismo con el estudio del Raspberry Pi ya que cuenta con características interesantes como la posibilidad de reproducir video en 1080p, la salida de video y audio a través de HDMI, conexión a Ethernet, puertos USB, además de varios sistemas operativos que se pueden implantar; inclusive se puede instalar Android y ganar múltiples funcionalidades desde contar con Google Play a utilizarlo como centro multimedia o servidor.

En el mercado existe una aplicación prototipo del *middleware* Ginga sobre la plataforma Android llamada *GingaMobile* que es dedicada a *smartphones*, la recomendación es integrar Ginga sobre este SO en la plataforma Raspberry Pi para obtener un sistemas más robusto.

De acuerdo al proceso de implementación y pruebas del *middleware* sobre el Raspberry Pi se recomienda utilizar el Ginga NCL desarrollado en Argentina, ya que brinda mayores facilidades de compilación; además de tener ayuda de la comunidad Ginga Argentino.

7.3. Trabajos Futuros

Como consecuencia de este proyecto de tesis se generan posibles trabajos en relación a la TV digital. A continuación se indican algunos temas:

- Trabajar con la integración del sintonizador y el *middleware*, teniendo una doble pantalla para uso de la TV digital y al mismo tiempo ejecutar aplicaciones, además la incorporación de control remoto para la interactividad.
- Evaluación y uso de Android, existe una ventaja al contar con Android ya que existe una infinidad de aplicaciones para su descarga, tanto de reproducción de juegos, música, etc. Es un sistema operativo muy versátil.
- Implementar un STB dedicado para recepción móvil (*one-seg*) que permita ver contenidos en dispositivos que se encuentren en movimiento, como equipos incorporados en buses y autos.
- Evaluación de la interactividad mediante integración de periféricos cámara, controles de juegos.

Anexos

UNIVERSIDAD DE CUENCA

desde 1867

Apéndice A

Características de algunos STB comerciales

A.1. STB comerciales

En la Tabla A.1 se muestra características de STB comerciales.

Nombre	**Sintonizador Tv Digital**	**FHD TV Decoder ISDB-T STB**	**Low-cost HD ISDB-T PVR STB**	**HD ISDB-T TV modulator FSeg**	**HD ISDB-T STB 1080p**	**RT STB Dual-core Android TVBox**	**Pro. Develop. ISDB-T Ginga TV STB**
Fabricante	Avanti	Chieta	Plute	EONTECH, OEM	Generico	Customised	Golden Stone
Modelo	STB-1638	ISDB-T 168mm	ISDB-T 803HD	DVB-2007IS	N/D	DV6801-T2	HIT224
Estándar	ISDB-Tb	ISDB-Tb	ISDB-Tb	ISDB-Tb	ISDB-Tb	ISDB-Tb	ISDB-Tb - Ginga-NCL
Señal de Entrada	Antenna in	Antenna in	Antenna in	Antenna in	Antenna in	Antenna in	Antenna in
Otras Entradas	USB, infrarojo, botón de encendido, botones de CH	USB, infrarojo, botón de encendido, botones de CH	USB, infrarojo, botón de encendido, de CH, de Menú y Play/Pause	USB, infrarojo, botón de encendido, botones de CH	USB, infrarojo, botón de encendido, botones de CH y de Volume	3xUSB, SD, Ethernet, infrarojo	USB, Ethernet, infrarojo
Salidas de Video	RCA (Video compuesto), HDMI (1080p), Antenna out	RCA (Video compuesto y por componentes), HDMI (1080p), Antenna out	RCA (Video compuesto y por componentes), S-Video, HDMI (1080p), Antenna out	RCA (Video compuesto, Video por componentes), HDMI (1080p), Antenna out	RCA (Video compuesto, Video por componentes), HDMI (1080p), Antenna out	HDMI (1080p), AV 3.5mm a RCA (Video compuesto), antenna out	HDMI (1080p), AV 3.5mm a RCA (Video compuesto), antenna out
Formato de Video	-	PAL / NTSC / Auto	PAL / NTSC / Auto	PAL / NTSC / Auto	-	-	-

Salida de Audio	RCA(L/R), coaxial S/PDIF	RCA(L/R)	RCA(L/R), coaxial S/PDIF	RCA(L/R)	RCA(L/R), coaxial S/PDIF	Optical S/PDIF, AV 3.5mm a RCA(L/R)	AV 3.5mm a RCA(L/R)
Otras Salidas	Pantalla LCD	Pantalla LCD	Pantalla LCD	LED encendido, LED signal lock	Pantalla LCD, LED encendido y signal lock	-	-
Conectividad	N/D	N/D	N/D	N/D	N/D	Ethernet 10M/100M, WiFi 2.4Ghz/5Ghz 802.11 b/g/n	Ethernet
Funciones	ISDB-Tb, Reproducción de Contenido de Usuario (USB)	ISDB-Tb, EPG, Teletexto y subtítulos, PVR, Reproducción de Contenido de Usuario (USB), Actualización de SW (USB), función de timer	ISDB-Tb, EPG, Teletexto y subtítulos, PVR, Reproducción de contenido de usuario (USB), Actualización de SW, Soporte 4:3 y 16:9	ISDB-Tb, EPG 7 días, Teletexto y subtítulos, PVR, Reproducción de contenido de usuario (USB), Actualización de SW, Soporte 4:3 y 16:9	ISDB-Tb, EPG 7 días, Teletexto y subtítulos, PVR, Reproducción de contenido de usuario (USB), Soporte 4:3 y 16:9	ISDB-Tb, EPG 7 días, Teletexto y subtítulos, PVR, Android 4.0, DLNA, Airplay, TVCast, Control Remoto mediante Smartphone, Soporte 4:3 y 16:9	ISDB-Tb, Ginga-NCL, STLinux 2.4

Especificaciones	-	Main chip: MSD7801, Tuner: MXL603	-	CPU: Mstar MSD7802	MSTAR-MSD7805 MIPS,64MB RAM, 32MB flash, TUNER Rafael R836	Amlogic AML8726-M3 ARM A9 1Ghz, Mali 400MP, 1GB RAM, 4GB flash, Tuner DIBCOM 8096	ST7102 650MHz, 4GB RAM, 2GB flash

Tabla A.1: Características de STB comerciales

Apéndice B

Instalaciones

B.1. Instalación del módulo de recepción ISDB-Tb

Como se mencionó en el Capítulo 5, en esta sección se van a detallar los pasos para instalar tanto el dispositivo Siano Mobile Silicon[1], como el dispositivo RTL2832U[2].

En primer lugar, se debe insertar el USB en uno de los puertos del Raspberry Pi, para comprobar que el equipo lo reconoce, se escribe el comando `lsusb`, donde a la salida se tendrá que verificar el modelo del USB. Para imprimir el nombre, versión y otros detalles de la máquina y el sistema operativo que se está ejecutando en ella, se usa el comando `uname -r`.

El siguiente paso es instalar las librerías necesarias:

```
$ sudo apt-get install git linux-headers-$(uname -r) build-essential
patchutils libproc-processtable-perl
```

Para sistemas operativos como el Raspbian, OSMC, Raspbmc, el sistema no reconoce el comando `linux-headers`, por lo que la construcción del kernel se tendrá que hacer paso a paso detallados en la siguiente sección.

El proyecto LinuxTV tiene la última serie de módulos de los controladores del kernel de Linux para dispositivos V4L-DVB, por lo cual se instala un paquete llamado: media_build, el cual maneja todos estos controladores, y que es necesario para el funcionamiento del sistema.

```
$ git clone git://linuxtv.org/media_build.git
```

[1] http://www.siano-ms.com/
[2] http://www.realtek.com.tw

```
$ cd media_build
$ ./build
$ sudo make install
```

B.2. Construcción de Kernel

Hay dos métodos principales para la construcción del kernel. Se puede construir localmente, que tendrá una duración de aproximadamente 10 horas; o se puede utilizar la compilación cruzada que es un método más rápido, y requiere de mayor configuración. A continuación se detalla los dos métodos:

B.2.0.1. Construcción del Kernel localmente

1. Obtener las fuentes
   ```
   $ git clone --depth = 1 https://github.com/raspberrypi/linux
   ```

2. Añadir las dependencias que falten:
   ```
   $ sudo apt-get install bc
   ```

3. Ejecutar el conjunto de comandos, Pi1 o Pi2 dependiendo de la versión de Raspberry Pi
   PI 1 Configuración por defecto
   ```
   $ cd Linux
   $ KERNEL = kernel
   $ make bcmrpi_defconfig
   ```

   PI 2 Configuración por defecto
   ```
   $ cd Linux
   $ KERNEL = kernel7
   $ make bcm2709_defconfig
   ```

4. Construir e instalar módulos
   ```
   $ make zImage modules dtbs
   $ sudo make modules_install
   $ sudo cp arch/arm/boot/dts/*.dtb /boot/
   $ sudo cp arch/arm/boot/dts/overlays/*.dtb* /boot/overlays/
   $ sudo cp arch/arm/boot/dts/overlays/README /boot/overlays/
   $ sudo scripts/mkknlimg arch/arm/boot/zImage /boot/$KERNEL.img
   ```

B.2.0.2. Compilación Cruzada

En primer lugar se va a requerir un Linux anfitrión. Se usa Ubuntu y se procede con los siguientes comandos:

1. Instalar cadena de herramientas
   ```
   $ git clone https://github.com/raspberrypi/tools
   ```

2. Obtener fuentes
   ```
   $ git clone --depth = 1 https://github.com/raspberrypi/linux
   ```

3. Generar fuentes, Pi1 o Pi2 dependiendo de la versión de Raspberry Pi
   PI 1 Configuración por defecto
   ```
   $ cd linux
   $ KERNEL = kernel
   $ make ARCH = CROSS_COMPILE arm = arm-linux-gnueabihf- bcmrpi_defconfig
   ```

   PI 2 Configuración por defecto
   ```
   $ cd linux
   $ KERNEL = kernel7
   $ make ARCH = CROSS_COMPILE arm = arm-linux-gnueabihf- bcm2709_defconfig
   ```

4. Construir módulos
   ```
   $ make ARCH = arm CROSS_COMPILE=arm-linux-gnueabihf- zImage modules dtbs
   ```


B.3. Compilación de VLC con aceleración de hardware

1. Instalar VLC desde el repositorio de Ubuntu Mate (y todos los paquetes que necesita).
   ```
   $ sudo apt-get install vlc browser-plugin-vlc
   ```

2. Obtener las herramientas para la compilación:
   ```
   $ sudo apt-get install git libtool build-essential pkg-config autoconf
   ```

3. Instalar las dependencias necesarias:
   ```
   $ sudo apt-get install liba52-0.7.4-dev libdirac-dev libdvdread-dev libkate
   -dev libass-dev libbluray-dev libcddb2-dev libdca-dev libfaad-dev libflac
   -dev libmad0-dev libmodplug-dev libmpcdec-dev libmpeg2-4-dev libogg-dev
   ```

```
libopencv-dev libpostproc-dev libshout3-dev libspeex-dev liblua5.1-0-dev
libopus-dev libschroedinger-dev libsmbclient-dev libtwolame-dev libx264-dev
 libspeexdsp-dev libssh2-1-dev libxcb-composite0-dev libxcb-randr0-dev
libxcb-xv0-dev libzvbi-dev libxcb-keysyms1-dev libsdl-image1.2-dev librsvg2
-dev libsamplerate0-dev libudev-dev libmtp-dev libupnp6-dev libnotify-dev
libdvbpsi-dev libgme-dev libebml-dev libgnomevfs2-dev libsidplay2-dev libva
-dev libjack-jackd2-dev libchromaprint-dev libxpm-dev libncurses5-dev
libsidplay1-dev libtar-dev libqt4-dev libncursesw5-dev
```

4. Descargar el más reciente código fuente VLC, configurar y compilar:

```
$ git clone git://git.videolan.org/vlc.git
$ cd vlc
$ export ACLOCAL_ARGS="-I /usr/share/aclocal"./bootstrap
$ ./configure --prefix=/usr --enable-rpi-omxil --disable-ogg
--disable-mux_ogg
$ make
```

5. La compilación durará aproximadamente 5 horas. Por último instalar:

```
$ sudo make install
```

6. Después de iniciar VLC, por primera vez, cambiar las siguientes opciones:
   Configuración Audio simple: ALSA y bcm2835.
   Configuración Video: OpenMAX.
   Overclocking RP: Ayuda especialmente para videos de alta definición. Esto dará más opciones y control que omxplayer.

B.4. Instalación de Ginga-NCL Brasil

Ginga-NCL es un entorno de presentación multimedia para aplicaciones declarativas escritas en NCL y su lenguaje de scripting Lua.

1. Obtener los paquetes necesarios:

```
$ sudo apt-get install build-essential automake autoconf subversion
libsysfs-dev libvncserver-dev liblzo2-dev x11-utils libx11-dev libpnglite
-dev libpng12-dev libjpeg62-dev libtiff4-dev libzip-dev libcurl4-openssl
-dev libcrypto++-dev libgpm-dev libexpat1-dev curl libxerces-c2-dev
libxerces-c3.1-devlibxext-dev libimlib2-dev libgmp-ocaml-dev libxmltooling
-dev git git-core -y
```

2. Exportar las variables a las dependencias de Ginga-NCL añadiendo las variables en el archivo `.bashrc` para no exportarlas en tiempo de ejecución, utilizamos el comando:
```
$ echo "export LD_LIBRARY_PATH=/usr/local/lib/lua/5.1/socket:/usr/local/lib
/ginga:/usr/local/lib/ginga/adapters:/usr/local/lib/ginga/cm:/usr/local/lib
/ginga/converters:/usr/local/lib/ginga/ic:/usr/local/lib/ginga/iocontents:
/usr/local/lib/ginga/players:/usr/local/lib/ginga/dp:/usr/local/lib/ginga/
mb:/usr/local/lib/ginga/mb/dec:/usr/local/lib64:/usr/local/lib:/usr/lib64:
/usr/lib:/lib64:/lib:/usr/kerberos/lib" >> ~/.bashrc
$ echo "export PKG_CONFIG_PATH=/usr/local/lib/pkgconfig" >> ~/.bashrc
```
Luego de exportar las variables es necesario, cerrar y abrir el terminal.

3. Crear un archivo `~/.directfbrc` con el siguiente contenido:
```
$ echo "export system=x11" >> ~/.directfbrc
$ echo "export mode=960x540" >> ~/.directfbrc
```

4. Instalar libtool:
```
$ sudo apt-get install libtool-bin
```

5. Instalar herramientas adecuadas y actualizadas para compilación:
```
$ sudo apt-get install g++  pkg-config cmake  cmake-dbg
```

6. Instalar librerías para LUA:
```
$ sudo apt-get install libluabind0.9.1 libluabind-doc libluabind-dev
libluabind-dbg libluabind-examples liblua5.1-curl-dev liblua5.1-curl0
liblua5.1-0-dev
```

7. Instalar librerías de codecs de video:
```
$ sudo apt-get install libxine1 libxine2 libxine-dev libxine2-dev
libavformat-dev libavcodec-dev libxine1-ffmpeg
$ sudo apt-get install  xine-ui
```

8. Instalar Faad2:
```
$ wget http://downloads.sourceforge.net/project/faac/faad2-src/faad2-2.7
/faad2-2.7.tar.gz
$ tar -zxvf faad2-2.7.tar.gz
$ cd faad2-2.7
$ ./configure --with-mp4v2
$ make
```

```
$ sudo make install
```

9. Instalar FFmpeg:

```
$  git clone git://source.ffmpeg.org/ffmpeg.git
$  cd ffmpeg
$ ./configure --enable-shared --enable-gpl --enable-nonfree --enable-pthreads
 --enable-postproc --disable-yasm
$ make
$ sudo make install
```

10. Instalar DirectFB:

```
$ sudo apt-get install libdirectfb-dev
```

A continuación descargar DirectFB[3] y descomprimir el archivo descargado. Ejecutar los siguientes comandos para su instalación:

```
$ cd DirectFB
$ ./configure --enable-x11 --with-gfxdrivers=none
$ make
$ sudo make install
```

11. Instalar DirectFB extra:

```
$ git clone git://git.directfb.org/git/directfb/extras/DirectFB-extra.git
$ cd DirectFB-extra
```

Dentro del paquete interfaces modificar los archivos `Makefile.am` de forma que todos los LIBADD contengan la variable `$(DFB_LIBS)`.

Por ejemplo: `libidirectfbvideoprovider_xine_la_LIBADD = $(DFB_LIBS) $(XINE_LIBS)`.

Luego de corregir los archivos necesarios se continúa con la instalación:

```
$ ./autogen.sh --disable-ffmpeg
```

Al final de los mensajes de salida es importante que Xine este activado: `Xine yes`.

```
$ make
$ sudo make install
```

12. Instalar DirectFB examples:

```
$ git clone git://git.directfb.org/git/directfb/extras/DirectFB-examples.git
$ cd DirectFB-examples
$ ./autogen.sh
$ make
$ sudo make install
```

Para verificar que DirectFB está funcionando se ejecuta los comandos:

[3] http://directfb.org/index.php?path=Main%2FDownloads

```
$ df_window
$ df_dok
$ df_andi
```

13. Instalar FusionSound:

```
$ git clone git://git.directfb.org/git/directfb/core/FusionSound.git
$ cd FusionSound
```

Antes de continuar se debe corregir el archivo `ifusionsoundmusicprovider_ffmpeg.c` ubicado en la ruta `/FusionSound/interfaces/IFusionSoundMusicProvider`. Las líneas a corregir son las siguientes, se muestra con un signo menos (-) la línea que debe ser remplazada por la que la precede el signo más (+):

```
- AVIOContext                    pb;
+ AVIOContext                    *pb;
- av_close_input_file( data->ctx );
+ avformat_close_input( &data->ctx);
- if (init_put_byte( &data->pb, data->iobuf, 4096, 0,
+ if ((data->pb = avio_alloc_context( data->iobuf, 4096, 0,
- if (av_open_input_stream(&data->ctx, &data->pb, filename, fmt, NULL)<0){
- D_ERROR("IFusionSoundMusicProvider_FFmpeg:av_open_input_stream() failed!\n");
+ if (avformat_open_input(&data->ctx, filename, fmt, NULL)<0) {
+ D_ERROR("IFusionSoundMusicProvider_FFmpeg:avformat_open_input() failed!\n");
- if (av_find_stream_info( data->ctx ) < 0) {
+ if (avformat_find_stream_info( data->ctx, NULL ) < 0) {
- if (!c || avcodec_open( data->codec, c ) < 0) {
+ if (!c || avcodec_open2( data->codec, c, NULL ) < 0) {
```

Luego de corregir las líneas necesarias se continúa con la instalación:

```
$ ./autogen.sh
$ make
$ sudo make install
```

14. Instalar LUA:

```
$ curl -R -O http://www.lua.org/ftp/lua-5.1.4.tar.gz
$ tar zxf lua-5.1.4.tar.gz
$ cd lua-5.1.4
```

Editar el archivo `src/Makefile`

Aumentar `-fPIC` en `CFLAGS`, ejemplo:

```
CFLAGS= -O2 -Wall -DLUA_COMPAT_ALL $(SYSCFLAGS) $(MYCFLAGS)
```

a

```
CFLAGS= -O2 -Wall -DLUA_COMPAT_ALL $(SYSCFLAGS) -fPIC $(MYCFLAGS)
$ make linux
$ sudo make linux install
```

15. Instalar LUA Socket:

```
$ wget http://luaforge.net/frs/download.php/2664/luasocket-2.0.2.tar.gz
$ tar -xvvzf luasocket-2.0.2.tar.gz
$ cd luasocket-2.0.2
$ make
$ sudo make install
```

16. Instalar Ginga-NCL

Descargar el código fuente de Ginga-NCL con el comando git

```
$ git clone http://git.telemidia.puc-rio.br/telemidia-util-cpp.git
$ git clone http://git.telemidia.puc-rio.br/telemidia-links-cpp.git
$ git clone http://git.telemidia.puc-rio.br/ncl30-cpp.git
$ git clone http://git.telemidia.puc-rio.br/gingacc-cpp.git
$ git clone http://git.telemidia.puc-rio.br/gingancl-cpp.git
$ git clone http://git.telemidia.puc-rio.br/gingalssm-cpp.git
$ git clone http://git.telemidia.puc-rio.br/ginga-cpp.git
```

La instalación normal de cada paquete se muestra a continuación :

Listado B.1: Instalación normal de paquetes

```
$ chmod 755 ./autogen.sh
$ ./autogen.sh
$ make
$ make install
```

Se procede a instalar cada uno de los paquetes:

`telemidia-util-cpp` Como en el Listado B.1.

`telemidia-links-cpp` El Makefile está configurado para una versión antigua de DirectFB. En versiones recientes la carpeta include fue trasladada de `/usr/include/` a `/usr/local/include/`. Crear un enlace para resolver el problema, con el siguiente comando:

```
$ sudo ln -s /usr/include/directfb /usr/local/include/directfb
```

Se instala las dependencias necesarias:

```
$ sudo apt-get install libgssapi-krb5-2 libkrb5-dev libadns1-dev
```

Y se compila como se muestra a continuación:

```
$ chmod 755 ./autogen.sh
$ ./autogen.sh --enable-graphics --with-directfb --enable-javascript
--without-x --without-sdl
$ make
$ make install
```

Ginga-cc

`gingacc-cpp/gingacc-system` Como en el Listado B.1.

`gingacc-cpp/gingacc-cm` Como en el Listado B.1.

`gingacc-cpp/gingacc-mb` Como en el Listado B.1.

`gingacc-cpp/gingacc-ic`

Se instala librerías necesarias con el siguiente comando:

`$ sudo apt-get install libfaad-dev mplayer` y se procede con la instalación del paquete como en el Listado B.1.

`gingacc-cpp/gingacc-um` Como en el Listado B.1.

`gingacc-cpp/gingacc-player` Se crea un enlace simbólico al archivo `core.so` con el nombre de `libcore.so`, dentro del directorio luasocket, por lo general está ubicado en `/usr/local/lib/lua/5.1/socket`.

Teniendo en cuenta el directorio anterior se utiliza el comando:

`$ sudo ln -s /usr/local/lib/lua/5.1/socket/core.so /usr/local/lib/lua/5.1/socket/libcore.so`

Y se procede con la instalación del paquete como en el Listado B.1.

`gingacc-cpp/gingacc-tuner` Se debe incluir la línea: `#include <unistd.h>` En los archivos `SocketProvider.h` y `UnicastProvider.cpp` ubicados en `/home/tesis/gingacc-cpp/gingacc-tuner/include/tuner/providers/` y `/home/tesis/gingacc-cpp/gingacc-tuner/src/providers/` respectivamente. Se procede con la instalación del paquete como en el Listado B.1.

`gingacc-cpp/gingacc-tsparser` Como en el Listado B.1.

`gingacc-cpp/gingacc-dataprocessing` Como en el Listado B.1.

`gingacc-cpp/gingacc-contextmanager` Como en el Listado B.1.

`gingacc-cpp/gingacc-multidevice` Como en el Listado B.1.

ncl30-cpp

`ncl30-cpp/ncl30` Como en el Listado B.1.

`ncl30-cpp/ncl30-converter` Como en el Listado B.1.

`gingancl-cpp` Como en el Listado B.1.

`gingalssm-cpp`

```
$ chmod 755 ./autogen.sh
$ ./autogen.sh --enable-tuner --enable-tsparser --enable-dataprocessing
$ make
$ sudo make install
```

`ginga-cpp` Como en el Listado B.1.

Una vez finalizada la instalación de Ginga-NLC se pueden correr los ejemplos con el siguiente comando:

```
$ ginga --ncl /ubicaciónArchivo/archivo.ncl
```

B.5. Instalación de Ginga.ar

Los pasos necesarios para compilar Ginga.ar en Linux son:

1. Obtener el código fuente

   ```
   $ apt-get update
   $ apt-get upgrade
   $ wget ftp://tvd.lifia.info.unlp.edu.ar/sources/Ginga.ar_2.1.zip
   $ unzip Ginga.ar_2.1.zip
   ```

2. Instalar las siguientes dependencias:

   Boost:

   ```
   $ sudo apt-get install libboost1.55-dev
    libboost-date-time1.55-dev
    libboost-date-time1.55.0
    libboost-serialization1.55-dev
    libboost-filesystem-dev
    libboost-filesystem1.55-dev
    libboost-filesystem1.55.0
    libboost-system-dev
    libboost-system1.55-dev
    libboost-system1.55.0
    libboost-math-dev
    libboost-math1.55-dev
    libboost-math1.55.0
    libboost-signals-dev
    libboost-signals1.55-dev
   ```

```
 libboost-signals1.55.0
 libboost-thread-dev
 libboost-thread1.55-dev
 libboost-thread1.55.0
```

XERCESC:

```
$ sudo apt-get install libxerces-c-dev
 libxerces-c3.1
```

EV:

```
$ sudo apt-get install libev-dev
 libev4
```

DB:

```
$ sudo apt-get install libdb-dev
 libdb5.3-dev
```

GTK2:

```
$ sudo apt-get install libgtk2.0-dev
```

GDK-PIXBUF:

```
$ sudo apt-get install libgdk-pixbuf2.0-dev
```

GLIB PANGO:

```
$ sudo apt-get install libglib2.0-dev
```

CAIRO:

```
$ sudo apt-get install libcairo-script-interpreter2
 libcairo2-dev
```

LIBSOUP2:

```
$ sudo apt-get install libsoup2.4-dev
```

WEBKIT:

```
$ sudo apt-get install libwebkitgtk-dev
 libwebkitgtk-1.0-0
 libwebkitgtk-1.0-common
```

LIBVLC:

```
$ sudo apt-get install libvlc-dev
 libvlc5
 libvlccore-dev
 libvlccore8
```

LUA:

```
$ sudo apt-get install liblua5.1-0
 liblua5.1-0-dev
 liblua5.1-socket2
```

CURL:

```
$ sudo apt-get install libcurl4-openssl-dev
```

GTEST:

```
$ sudo apt-get install libgtest-dev
```

CMAKE:

```
$ sudo apt-get install cmake
```

Se recomienda instalar la versión 2.8.7 de CMake con los siguientes pasos:

Verificar la versión instalada de cmake con: `$ --version`

Descargar la versión 2.8.7 del sitio web y descomprimir la carpeta.

Crear un directorio `_build` en la carpeta: `mkdir_build`

En el directorio `_build`, ejecutar los siguientes comandos para construir e instalar los fuentes de CMake:

```
$ cmake .. -DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=/usr
$ make
$ sudo make install
$ sudo ldconfig
```

Compilar la dependecia de gtest, con los siguientes comandos:

```
$ cd /usr/src/gtest
$ sudo cmake . - sudo cmake  CMakeLists.txt
$ sudo make
```

Al hacer sudo make: No targets specified and no makefile found

```
$ sudo mv libg* /usr/lib/  - sudo cp *.a /usr/lib
```

3. Construir las fuentes, dentro de la carpeta donde se descomprimieron:

```
$ mkdir install
$ cd install
$ ../build/build.py -t ginga
```

4. Instalar VLC:

```
$ sudo apt-get install vlc browser-plugin-vlc
```

Finalizada la compilación en `UNIX/bin/ginga` se encuentra el binario de Ginga.ar. Para ejecutar las aplicaciones se utiliza el comando:

```
./ginga --ncl /aplicacion.ncl
```

Apéndice C

Archivo de canales para el reproductor VLC

C.1. Archivo canales.xspf

```
<?xml version="1.0" encoding="UTF-8"?>
<playlist xmlns="http://xspf.org/ns/0/" xmlns:vlc="http://www.
    videolan.org/vlc/playlist/ns/0/" version="1">
        <title>DVB Playlist</title>
        <creator>w_scan-20140727</creator>
        <info>http://wirbel.htpc-forum.de</info>
        <trackList>
                <track>
                        <title>0001. Avalpa1  MPEG2 MHP</title>
                        <location>dvb-t://frequency=557142857</
                            location>
                        <extension application="http://www.videolan.
                            org/vlc/playlist/0">
                                <vlc:option>dvb-bandwidth=6</
                                    vlc:option>
                                <vlc:option>dvb-ts-id=1</vlc:option>
                                <vlc:id>2</vlc:id>
                                <vlc:option>program=1</vlc:option>
                        </extension>
```

```
        </track>
        <track>
                <title>0002. Avalpa2  MPEG2 MHEG5</title>
                <location>dvb-t://frequency=557142857</
                    location>
                <extension application="http://www.videolan.
                    org/vlc/playlist/0">
                        <vlc:option>dvb-bandwidth=6</
                            vlc:option>
                        <vlc:option>dvb-ts-id=1</vlc:option>
                        <vlc:id>3</vlc:id>
                        <vlc:option>program=2</vlc:option>
                </extension>
        </track>
        <track>
                <title>0003. Avalpa3  MPEG2 HBBTV</title>
                <location>dvb-t://frequency=557142857</
                    location>
                <extension application="http://www.videolan.
                    org/vlc/playlist/0">
                        <vlc:option>dvb-bandwidth=6</
                            vlc:option>
                        <vlc:option>dvb-ts-id=1</vlc:option>
                        <vlc:id>4</vlc:id>
                        <vlc:option>program=3</vlc:option>
                </extension>
        </track>
        <track>
                <title>0004. Avalpa4  MPEG2 TXT</title>
                <location>dvb-t://frequency=557142857</
                    location>
                <extension application="http://www.videolan.
                    org/vlc/playlist/0">
                        <vlc:option>dvb-bandwidth=6</
                            vlc:option>
                        <vlc:option>dvb-ts-id=1</vlc:option>
                        <vlc:id>5</vlc:id>
                        <vlc:option>program=4</vlc:option>
                </extension>
```

```
        </track>
        <track>
                <title>0005. Avalpa5  H264</title>
                <location>dvb-t://frequency=557142857</
                    location>
                <extension application="http://www.videolan.
                    org/vlc/playlist/0">
                        <vlc:option>dvb-bandwidth=6</
                            vlc:option>
                        <vlc:option>dvb-ts-id=1</vlc:option>
                        <vlc:id>6</vlc:id>
                        <vlc:option>program=5</vlc:option>
                </extension>
        </track>
        <track>
                <title>0006. Avalpa6  HD H264</title>
                <location>dvb-t://frequency=557142857</
                    location>
                <extension application="http://www.videolan.
                    org/vlc/playlist/0">
                        <vlc:option>dvb-bandwidth=6</
                            vlc:option>
                        <vlc:option>dvb-ts-id=1</vlc:option>
                        <vlc:id>7</vlc:id>
                        <vlc:option>program=6</vlc:option>
                </extension>
        </track>
        <track>
                <title>0007. Ecuador SD1</title>
                <location>dvb-t://frequency=671142857</
                    location>
                <extension application="http://www.videolan.
                    org/vlc/playlist/0">
                        <vlc:option>dvb-bandwidth=6</
                            vlc:option>
                        <vlc:option>dvb-ts-id=1854</
                            vlc:option>
                        <vlc:id>8</vlc:id>
```

```
                        <vlc:option>program=59328</
                            vlc:option>
                    </extension>
                </track>
                <track>
                    <title>0008. Ecuador SD2</title>
                    <location>dvb-t://frequency=671142857</
                        location>
                    <extension application="http://www.videolan.
                        org/vlc/playlist/0">
                        <vlc:option>dvb-bandwidth=6</
                            vlc:option>
                        <vlc:option>dvb-ts-id=1854</
                            vlc:option>
                        <vlc:id>9</vlc:id>
                        <vlc:option>program=59329</
                            vlc:option>
                    </extension>
                </track>
                <track>
                    <title>0009. Ecuador 1seg</title>
                    <location>dvb-t://frequency=671142857</
                        location>
                    <extension application="http://www.videolan.
                        org/vlc/playlist/0">
                        <vlc:option>dvb-bandwidth=6</
                            vlc:option>
                        <vlc:option>dvb-ts-id=1854</
                            vlc:option>
                        <vlc:id>10</vlc:id>
                        <vlc:option>program=59352</
                            vlc:option>
                    </extension>
                </track>
        </trackList>
</playlist>
```

Listado C.1: Archivo canales.xspf para VLC

Bibliografía

[1] L. A. Venegas, "Generación de una trama broadcast transport stream (bts) usando el software libre opencaster," 5 2012.

[2] V. J. Marsola, "Conversor de TV Digital Terrestre: Set-top box," TELECO, Tech. Rep., 2008. [En línea]. Disponible: http://www.teleco.com.br/pdfs/tutorialconvtvd.pdf.pdf

[3] J. L. V. Melo, "Diseño y desarrollo de aplicaciones interactivas para el middleware ginga de televisión digital de la norma isdb-tb para brindar información de los protocolos de prevención a la población en lugares de alto riesgo de erupciones volcánicas, sismos y tsunami," 4 2013.

[4] UNED, "Electrónica para sistemas industriales," UNED, Tech. Rep. [En línea]. Disponible: http://www.ieec.uned.es/investigacion/Dipseil/PAC/archivos/Informacion_de_referencia_ISE3_3_1.pdf

[5] P. oficial Arduino, "Arduino Mega 2560 R3." [En línea]. Disponible: http://arduino.cl/arduino-mega-2560/

[6] Página oficial raspare pi, "Productos." [En línea]. Disponible: https://www.raspberrypi.org/products/

[7] P. oficial BeagleBone, "Beaglebone." [En línea]. Disponible: http://beagleboard.org/bone

[8] internetdelascosas.cl, "Beaglebone." [En línea]. Disponible: http://www.internetdelascosas.cl/2013/05/20/beaglebone-black-primeros-pasos/

[9] J. M. y Cristian Villa, "Implementación de un laboratorio de televisión digital que cumpla el estándar isdb-tb," 6 2014.

[10] A. G. y Miguel Cochancela, "Diseño de un laboratorio de televisión digital para la transmisión de señales con multiprogramación, contenidos interactivos y guía electrónica de programación," 4 2013.

[11] Francisco Sandoval, "Sistema ISDB-T e ISDBT-Tb, Integrated Service Digital Broadcasting Terrestrial," Tech. Rep. [En línea]. Disponible: http://reftelinteractivadigital.blogspot.com/2015/09/isdb-t-e-isdb-tb.html

[12] T. Svangard, "An evaluation of the pandaboard as a set-top-box in an android environment," Master's thesis, UPPSALA UNIVERSITET, 12 2011.

[13] Y. Wahyu, F. Oktafiani, y Y. Saputera, "Development of set top box (stb) for dvb-t2 standard television based on android," in *Telecommunication Systems Services and Applications (TSSA), 2014 8th International Conference on*, Oct 2014, pp. 1–4.

[14] P. oficial Arduino, "Introducción." [En línea]. Disponible: https://www.arduino.cc/

[15] M. P. González, "Estudio piloto de los demoduladores de la serie rtl de realtek para la radio definida por software," Master's thesis, Universidad de Sevilla, 1 2014.

[16] Radioafición.com, "RTL2832U." [En línea]. Disponible: http://www.radioaficion.com/HamNews/articles/13247-sdr-barato-y-facil.html

[17] Siano, "Advanced Receiver for DVB-T2, DVB-T, ISDB-T and Digital/FM Radio Superior Solution for Automotive and CE Applications ," Siano, Tech. Rep. [En línea]. Disponible: http://www.siano-ms.com

[18] C. G. Ecuador, "Comunidad Ginga Ecuador." [En línea]. Disponible: http://comunidadgingaec.blogspot.com/

[19] MINTEL, "Proceso de Implementación de la Televisión Digital en Ecuador," MINTEL, Tech. Rep., 2015. [En línea]. Disponible: http://www.telecomunicaciones.gob.ec/wp-content/uploads/downloads/2015/02/PRESENTACIO%CC%81N_TDT_MINTEL-Febrero-2015.pdf

[20] CONATEL, "Resolución 084-05-CONATEL-2010-Adopción del estándar ISDB-Tb," 2014. [En línea]. Disponible: http://www.arcotel.gob.ec/wp-content/uploads/downloads/2013/07/084_05_conatel_2010.pdf

[21] INEN, " Televisores con sintonizador del estándar de televisión digital ISDB-T Internacional," 2013. [En línea]. Disponible: http://www.normalizacion.gob.ec

[22] ABNT, *ABNT NBR 15601 - Televisión Digital Terrestre - Sistema de transmisión ISDB-Tb*, ABNT Std., Rev. 30.11.2007, 08 2007.

[23] J. A. P. Albán, "Estudio técnico, económico y legal para la implementación práctica del canal de televisión de televisión digital terrestre ecotel-tv de la ciudad de loja, bajo el estándar isdb-tb/sbtvd adoptado en el ecuador," 5 2014.

[24] SENATEL, "INFORME CITDT-GATR-2012-005-Metodología de asignación de frecuencias temporales para la operación de estaciones de televisión digital terrestre. ," SENATEL, Tech. Rep., 2012. [En línea]. Disponible: http://www.telecomunicaciones.gob.ec/wp-content/uploads/downloads/2013/02/Informe-CITDT-GATR-2012-005.pdf

[25] Ruymán Ojeda García, "Set-Top Box- Microprocesadores para Comunicaciones." [En línea]. Disponible: http://www.iuma.ulpgc.es/~nunez/clases-micros-para-com/mpc0809-trabajos/mpc0809RuymanOjedaSTBs.pdf

[26] S. Pravin y R. Balakrishnan, "Set top box system with android support using embedded linux operating systempaper," in *Advances in Engineering, Science and Management (ICAESM), 2012 International Conference on*, March 2012, pp. 474–478.

[27] G. Calixto, C. Hira, L. Costa, y R. de Deus Lopes, "An open source and low cost solution for consumer electronics middleware validation," in *Consumer Electronics (ISCE), 2013 IEEE 17th International Symposium on*, June 2013, pp. 159–160.

[28] B. Toledo Freitas, A. Susin, y A. Bonatto, "Ginga middleware on a soc for digital television set-top box," in *Circuits and Systems (LASCAS), 2014 IEEE 5th Latin American Symposium on*, Febrero 2014, pp. 1–4.

[29] V. autores, "Ginga-NCL Conformance Testing." [En línea]. Disponible: http://testsuite.gingancl.org.br

[30] P. oficial Ubuntu Mate, "Ubuntu Mate." [En línea]. Disponible: https://ubuntu-mate.org/

[31] Elonics, *Multi-Standard CMOS Terrestrial RF Tuner*, 2010.

[32] D. Phil, *Realtek RTL2832U The mystery chip at the heart of RTL-SDR.*, 2015.

[33] A. DÁgostino, "Redusers te explica la interactividad en la TV digital." [En línea]. Disponible: http://www.redusers.com/noticias/redusers-te-explica-de-que-va-la-interactividad-en-la-tv-digital/

[34] Irina B. Siles, Pedro J. Orbea, Rafael Oliveira, Héctor Cruz Enríquez, "Desarrollo de la aplicación BEISBOL.HIP para la TDT," Tech. Rep., 2014.

[35] ARIB, *ARIB STD-B31 V1.6- Transmission System for Digital Terrestrial Television Broadcasting*, ARIB Std., Rev. 30.11.2005, 11 2005.

[36] ARIB_, *ARIB STD-B21 V4.6- Receiver for Digital Broadcasting*, ARIB_ Std., Rev. 14.03.2007, 03 2007.

[37] Comunidad ginga brasil, "Sitio Oficial del Middleware Ginga," 2010. [En línea]. Disponible: http://www.ginga.org.br/

[38] Lifia, "Ginga.ar." [En línea]. Disponible: http://tvd.lifia.info.unlp.edu.ar/ginga.ar/index.php/download

[39] ABNT, "Asociación Brasileña de Normas Técnicas." [En línea]. Disponible: http://www.abnt.org.br/

[40] Luis Fernando Gomes Soares, "TV Interactiva se hace con Ginga," Universidad Católica Río de Janeiro, Tech. Rep. [En línea]. Disponible: http://www.telemidia.puc-rio.br/sites/telemidia.puc-rio.br/files/2009_04_soares.pdf

[41] C. y. T. Ministerio de Educación Universitaria, "Televisión Digital Abierta." [En línea]. Disponible: http://tda.cnti.gob.ve/